National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

University of Alberta

Design, Construction, and Adaptive Control of a Flexible Manipulator

by

Frantisek Niscak

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDY AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

Doctor of Philosophy

Department of Electrical Engineering

Edmonton, Alberta

Fall, 1991

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-70001-7

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Frantisek Niscak

TITLE OF THESIS: Design, Construction, and Adaptive Control of a Flexible Manipulator

DEGREE: Doctor of Philosophy

YEAR THIS DEGREE GRANTED: Fall, 1991

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

SIGNED: F. Niscak

ADDRESS: 5176 Del Monte Avenue

Victoria, B.C., V8Y 1X2

DATE: October 8, 1991

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Design, Construction, and Adaptive Control of a Flexible Manipulator**, submitted by Frantisek Niscak in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

V. G. Gourishankar

Prof. V. G. Gourishankar (Supervisor)

Prof. R. E. Rink

Prof. S. L. Shah

Roger W Toogood

Prof. R. W. Toogood

Prof. Y. J. Kingma

Prof. M. M. Gupta (External Examiner)

Date: October 18 1991

To the Memory of My Parents

Abstract

This thesis deals with the design, construction, and control of a flexible manipulator. Since the objective was to implement control algorithms on an actual system, a test-bed, consisting of the manipulator, hardware and software subsystems was designed and built. Also the subsystems for interfacing, power distribution, brakes, signal filtering, sensors etc. are parts of the constructed test-bed. This test-bed can be used for future research in the control of flexible manipulators.

The 2 DOF manipulator consists of two *megatorque* direct-drive motors, one rigid and one flexible link. The computer system controlling the manipulator is based on 25 MHz 80386 CPU and a parallel processing environment consisting of four 25 MHz T800 transputers. The simulations and actual experiments in this thesis will be confined to planar motion only.

Since a good analytical model of the 2 DOF flexible manipulator is not available due to nonlinearities, unknown inertias, motor torque-speed characteristics etc. two control strategies which can cope with this situation are developed: the adaptive multi-input multi-output generalized predictive control (MIMO GPC) algorithm and the multivariable PID control algorithm. Before implementing the MIMO GPC on the actual manipulator, simulation studies were carried out.

Although the control requirement is to control the position of the tip of the flexible link, it is not directly possible to do so with currently available position sensors. Instead, the motor shaft position and the tip acceleration are measured and controlled. When the motor position arrives at its final position and the tip acceleration is damped out, it is concluded that the tip is at the desired final position. The adaptive MIMO GPC was very effective in controlling the motor shaft position whereas better control of the flexible link vibrations was achieved when the multivariable PID control algorithm was used.

Computational demands dictated by the dimensionality and complexity of the adaptive MIMO GPC led to inclusion of a transputer based parallel processing environment.

ACKNOWLEDGEMENT

I would like to thank to my advisor, Professor V.G. Gourishankar, whose constant guidance, patience, and support throughout my graduate studies at the University of Alberta were invaluable. I am greatly indebted to my co-supervisor, Professor R.E. Rink, for his encouragement and advice over the course of this research. I would especially thank Professor S. Shah for sharing his expertise in the adaptive generalized predictive algorithm and Professor R.W. Toogood for his guidance in robotics.

I highly value Professor Y.J. Kingma's inspiration and engineering advice on hardware aspects of this project. I would also like to express my appreciation to him for allowing me to use his lab facilities and equipment. Warm thanks go to Dave Rudyk for sharing his skill and practical advice with hardware construction. My thanks and appreciation for help with mechanical construction of this project go to my friend Herb Gans, machine shop supervisor, and his staff in the machine shop.

I am grateful to all staff members and Professor P.R. Smy, chairman, of the Department of Electrical Engineering for providing an environment conducive to research.

My research would not have been possible without the generous financial support from NSERC and Killam Trust. Additional funding was provided by my supervisors and the Department of Electrical Engineering.

During my graduate studies I have gained many friends who greatly enriched my life with their friendship. I would like to thank Ury Ravid and his family for lasting friendship. Ury's optimism was a constant driving force. My project was enriched by technical discussions with my friend Dr. J.O. King. I will always remember the flashing smile of my friend Philip Pang with whom I have shared many long evenings in the robotics lab.

I will never forget my days in Professor Y.J. Kingma's lab where I was lucky to share the companionship of my close friend Con Verburg. Our tennis games, jogging, and long lab hours will not be forgotten. Thank you, Con.

I would like to thank to my sister Alena, brothers Vaclav and Jaromir, and my son David as well as to my newly acquired family, Daniel, Kathleen, and Ellen, for their patience and understanding during the long years of this research.

The quality of my life during this project was enriched by my wife Sharon whose warm companionship, patience, constant optimism, and support can never be appreciated enough. Words cannot express my feelings and love.

My endless thanks go to all of you mentioned and not mentioned.

Contents

1 Introduction 1

1.1 Introduction . . . 1

1.2 Literature Review . . . 2

1.2.1 Rigid Manipulators . . . 3

1.2.2 Manipulator Design . . . 6

1.2.3 Adaptive Control of Manipulators . . . 9

1.2.4 Experimental Research of Flexible Manipulators . . . 11

1.3 Contributions of this Thesis . . . 15

1.4 Organization of the Thesis . . . 15

2 Manipulator Design 17

2.1 Introduction . . . 17

2.2 Configuration and Task Specification . . . 18

2.3 Trajectory Considerations . . . 18

2.4 Manipulator Kinematics . . . 26

2.5 Torque Calculations . . . 29

2.6 Choice of Actuators . . . 32

2.7 Link Choice . . . 37

3 Manipulator Hardware Design 41

3.1 Introduction . . . 41

3.2 Quadrature Decoder Board . . . 43

3.3 Digital I/O Interface . . . 46

3.4 Strain Gages and Strain Gage Amplifiers . . . 49

3.5 3 Phase Power Distribution and Motor Brake Activation . . . 52

3.6 Antialiasing Filters . . . 52

3.7 Double Integrator . . . 55

3.8 Software Drivers . . . 57

3.8.1 Hardware Timer Interrupt 58
3.8.2 Analog to Digital Conversion 58
3.8.3 Motor Shaft Position 58

4 Assessment of Test-bed Performance **59**
4.1 Introduction 59
4.2 Friction 61
4.3 Performance of PID Controller with Different Friction Compensation Schemes 67
4.4 Combined Coulomb and Viscous Frictions 70
4.5 DD Motor Dynamics 73
4.5.1 Motor Delays and Rise Time $t_{\omega max}$ 73
4.5.2 Open-Loop Frequency Response 73
4.6 Determination of the Accelerometer Range 76
4.7 Factors Affecting the Double Integrator Performance 79
4.8 Measurements of Flexible Link Frequencies 84
4.8.1 Clamped-Free Modal Frequencies 84
4.8.2 Link Frequencies in Closed-loop Configuration 85

5 Adaptive MIMO GPC - Theory and Some Implementation Issues **89**
5.1 Introduction 89
5.2 SISO GPC 90
5.3 MIMO GPC 96
5.4 Free and Forced Responses for MIMO GPC 102
5.5 Input and Output Filtering for the Free Response 109
5.6 MIMO GPC Algorithm - Some Implementation Issues 112
5.7 Matrix Inverse and Control Signal Calculations 116
5.8 Control Constraints 120
5.9 Output Error Weighting Coefficients 121
5.10 Nonsquare Systems and MIMO GPC 123
5.11 T Filter 123
5.12 Identification Procedure for MIMO GPC 125

6 Simulations **132**
6.1 Introduction 132
6.2 Smooth Set-point Transition 133
6.3 Output Decoupled Nonsquare Plant 135

6.4 Output Coupled Nonsquare Plant 147
6.5 Summary 154

7 Experimental Results **161**
7.1 Introduction 161
7.2 Adaptive MIMO GPC 162
7.3 Multivariable PID Control Law 168
7.4 Summary 176

8 Parallel Processing and Some Software Implementation Issues **180**
8.1 Introduction 180
8.2 Timing Requirements 181
8.3 Choice of the Parallel Processing Environment - Transputer 182
8.4 Configuration and Software Support 184
8.5 Message Passing Protocol 187
8.6 MIMO GPC Parallelization 191
8.7 Debugging and Error Detection 195
8.8 Additional Speed Considerations 196
8.9 Algorithm Transformations 198

9 Conclusion **200**
9.1 Summary 200
9.2 Recommendation for Future Research 202

Bibliography **203**

A Hardware Design **214**
A.1 QDB 214
A.1.1 Quadrature Signal Decoder and Counter Subsystems 214
A.1.2 Sampled Counter Memory and I/O Bus Drivers 217
A.1.3 Quadrature Board Controller 218
A.2 DI/O Box 219
A.3 Strain Gage Amplifiers 219
A.4 4-th Order Bessel Antialiasing Filters 220
A.5 Design of the Double Integrator 226
A.6 Hardware Timer Interrupt Generation 229
A.7 ADC Software driver 230
A.8 QDB Software Driver 231

B Software Drivers - Source Code **232**

B.1 Tecmar.c 232

B.2 Encoder.asm 236

B.3 IRQ5.asm 237

B.4 ADC.asm 239

C Schematic Diagrams - Orcad **241**

List of Figures

2.1 Manipulator configuration. 19
2.2 Manipulator workspace. 19
2.3 Piece-wise linear velocity profile. 20
2.4 Quasi-trapezoidal velocity profile. 21
2.5 Kinematic quantities with zero boundary jerk. 23
2.6 Kinematic quantities with nonzero boundary jerk. 25
2.7 2 DOF robotics manipulator. 27
2.8 Torques for the 2 DOF robotics manipulator. 30
2.9 Assembly of the 2 DOF manipulator. 35
2.10 2 DOF manipulator and acceleration signal conditioning unit. 36

3.1 2 DOF robotics manipulator hardware diagram. 42
3.2 Block diagram of the QDB PC board. 45
3.3 Quadrature decoder board (component side). 47
3.4 Quadrature decoder board (wire-wrapped side). 48
3.5 Digital I/O box and its power supplies. 50
3.6 Strain gage amplifiers and power supply. 51
3.7 Brake control system (top), 3 phase distribution (middle), and direct-drive motor amplifiers (bottom) - front view. 53
3.8 Brake control system (top), 3 phase distribution (middle), and direct-drive motor amplifiers (bottom) - rear view. 54
3.9 Bessel filter and double integrator box. The 4-th order Bessel filter board (bottom right) and the double integrator board (bottom left). 56
3.10 Double integrator. 57

4.1 Complete test-bed. 60
4.2 Initial step response of the base motor with a PID controller. 61
4.3 Stiction for counterclockwise rotation - two experiments. 64
4.4 Stiction for clockwise rotation - two experiments. 64

4.5 Comparison of stiction in both directions. 65
4.6 Step response with uncompensated PID controller. 68
4.7 Step response of a PID controller with Coulomb friction compensation. 69
4.8 Step response of a PID controller with stiction compensation. 69
4.9 Step response of a PID controller with stiction and Coulomb friction compensation. 70
4.10 Characteristics based on measured velocity. 72
4.11 Characteristics based on measured position. 72
4.12 Measured acceleration from a closed-loop experiment. 78
4.13 Complete discharge - an example. 81
4.14 Incomplete discharge - an example. 83
4.15 Acceleration and its power spectral density when only 2 modal frequencies are excited. 85
4.16 Acceleration and its power spectral density when the first 3 modal frequencies are excited. 86
4.17 Applied control voltage and its power spectral density during a closed-loop experiment with a PID controller. 87
4.18 Flexible link acceleration and its frequency spectrum during a closed-loop experiment with a PID controller. 88

6.1 Output and control signals for a double integrator plant with step changes in set-point, $NU = 2$. 134
6.2 Output and control signals for a double integrator plant with sinusoidal transitions in set-point, $NU = 2$. 135
6.3 Simulation Run 1. 138
6.4 Simulation Run 2. 139
6.5 Simulation Run 3. 140
6.6 Simulation Run 4. 141
6.7 Simulation Run 5. 143
6.8 Simulation Run 6. 144
6.9 Simulation Run 7. 145
6.10 Simulation Run 8. 146
6.11 Simulation Run 9. 148
6.12 Simulation Run 10. 150
6.13 Simulation Run 11. 151
6.14 Simulation Run 12. 152

6.15 Simulation Run 13. 153
6.16 Simulation Run 14. 155
6.17 Simulation Run 15. 156
6.18 Simulation Run 16. 157
6.19 Simulation Run 17. 158
6.20 Simulation Run 18. 159

7.1 GPC: Experimental Run 1. 164
7.2 Identified $A_{22}(q^{-1})$ polynomial. 164
7.3 GPC: Experimental Run 2. 165
7.4 GPC: Experimental Run 3. 166
7.5 GPC: Experimental Run 4. 166
7.6 GPC: Experimental Run 5. 167
7.7 GPC: Experimental Run 6. 168
7.8 GPC: Experimental Run 7. 169
7.9 GPC: Experimental Run 8. 169
7.10 Multivariable PID controller - block diagram. 170
7.11 PID: Experimental Run 1. 172
7.12 PID: Experimental Run 2. 173
7.13 PID: Experimental Run 3. 173
7.14 PID: Experimental Run 4. 174
7.15 PID: Experimental Run 5. 175
7.16 PID: Experimental Run 6. 175
7.17 PID: Experimental Run 7. 176
7.18 PID: Experimental Run 8. 177
7.19 Acceleration and its power spectral density when the flexible link vibrates vertically but measurements are in the horizontal direction. . . 179

8.1 Parallel processing hardware configuration. 185
8.2 Examples of task allocation for 1D motion. 189
8.3 Example of task allocation for 2D motion. 190
8.4 1D adaptive GPC with strict timing requirements. 193
8.5 1D adaptive GPC with relaxed timing requirements. 194
8.6 Adaptive GPC for decoupled 2D motion. 194
8.7 Adaptive GPC for coupled 2D motion. 195

A.1 Quadrature signals. 215

A.2 Karnaugh map for the QDB. 216
A.3 Amplitude characteristic for 4-th order Bessel filter. 221
A.4 Phase characteristic of 4-th order Bessel filter. 222
A.5 Type 2B low-pass filter. 223
A.6 Schematic diagram for discharging the double integrator. 228

C.1 Logic for brakes and 3 phase power. 242
C.2 Block diagram of the QDB. 243
C.3 Interconnections among basic QDB subsystems. 244
C.4 Prefilter and decoder. 245
C.5 6-bit up/down Counter - EPLD library. 246
C.6 24-bit up/down counter - EPLD900. 247
C.7 Counter memory and I/O bus drivers - Part 1. 248
C.8 Counter memory and I/O bus drivers - Part 2. 249
C.9 Counter memory and I/O bus drivers - Part 3. 250
C.10 QDB controller - internal structure. 251
C.11 QDB controller - external structure. 252
C.12 Interface for CN2 inputs. 253
C.13 Interface for CN5 inputs. 254
C.14 Interface for CN2 outputs. 255
C.15 Computer side power supply for the DI/O box. 256
C.16 Motor side power supply for the DI/O box. 257
C.17 Strain gage amplifier. 258
C.18 Power supply for the strain gage amplifier. 259
C.19 The 4-th order Bessel antialiasing filter. 260
C.20 The 4-th order Bessel filter with component carier. 261
C.21 Double integrator. 262

List of Tables

2.1 Velocity and acceleration for the base motor. 26
2.2 Velocity and acceleration for the shoulder motor. 26
2.3 Modal frequencies for clamped-free and pinned-free vibrations. 40

4.1 Average control voltage to overcome stiction in both directions. . . . 66
4.2 Average control voltage corresponding to Coulomb friction in both directions. 66
4.3 Measured delays and rise time. 74
4.4 Amplitude response to a sinusoidal input. 76
4.5 Phase shift of the response to a sinusoidal input. 76
4.6 Acceleration from open-loop sinusoidal experiments. 77
4.7 Maximum accelerations from open-loop step measurements. 78
4.8 Power spectral density of the control input signal. 86
4.9 Acceleration power spectral densities. 86

5.1 Corner frequencies [rad/sec]. 125

8.1 MIMO GPC timing. 183
8.2 MIMO RPEM timing. 183

Chapter 1

Introduction

1.1 Introduction

Although structural flexibility is inherent in all mechanical structures, the majority of the current industrial robotic manipulators are classified as rigid manipulators and every effort is made to minimize flexibility effects. Links of such manipulators are heavy and their length to thickness ratio is relatively low. The resulting mechanical manipulator structures then exhibit only vibrations of small amplitudes and high frequencies which do not present a serious problem of control.

However, rigid manipulators must pay a price for their bulkiness. Their actuators must be capable of delivering large torques to move payloads whose mass is only a small fraction of the manipulator mass and the achieved speeds are low. Therefore, rigid manipulators mean higher energy consumption, large manipulator mass and inertia, and lower achievable velocities. On the positive side, control algorithms for the rigid manipulators are less complicated than the control algorithms which take flexibility effects into account.

It is desirable to have low weight manipulators which are reasonably priced and at the same time possess relatively accurate trajectory-following characteristic.

Book and Majette [17] characterized the main advantages of using the light weight flexible manipulators as follows:

- Smaller actuators
- Lower energy consumption
- Lower cost
- Higher speed

- Safer operations due to reduced inertia
- Better back-driveability due to elimination of gearing mechanisms
- Lower mass of the manipulator (important for space applications)

However, before there is a demand for such manipulators, work must be done to develop suitable control algorithms which are computationally feasible. The currently proposed control algorithms frequently require good models for the flexible dynamics, more powerful computational facilities, better sensors, and improved robustness. Such research effort is also warranted by the already established need for lightweight manipulators in space applications. There is also a potential use for such manipulators in certain assembly line operations.

Initial research was mostly simulation based and only later some experimental studies appeared, some of which are surveyed in the following section. However, researchers such as Tomizuka [100] and Kotnik et al. [55] have pointed out that verification of control algorithms for robotics applications via simulations is not sufficient and that it is very likely that many of the proposed algorithms would fail if they were applied to real robotic manipulators.

The dynamics of manipulator motions are highly nonlinear and coupled even in the case of rigid manipulators and inclusion of flexibility effects increases this complexity. Additional nonlinearities are introduced by the actuators (such as D.C. motors) since a real actuator exhibits saturation characteristics as well as other nonlinearities due to Coulomb and viscous friction. Dead-zone characteristic of a tachometer should be taken into account as well.

1.2 Literature Review

The field of research in robotics control is quite diverse and consists of many different topics. It is not possible, nor is it necessary, to undertake a review of the literature covering the entire area. In this section the following topics are covered.

In subsection 1.2.1 a brief survey, highlighting important results in the control of rigid manipulators, is given. Although this thesis is concerned with flexible manipulators, the techniques used for flexible manipulator control often use ideas and concepts from the rigid manipulator control literature.

Literature related to manipulator design is surveyed in subsection 1.2.2. Since the major portion of the project reported in this thesis focuses on the design and

construction of an actual 2 DOF manipulator (with provision for a third link) with a complete computer system, a number of papers related to manipulator design will be surveyed.

Adaptive control techniques for both rigid and flexible manipulators are surveyed in subsection 1.2.3.

Literature related to experimental research (actual implementation) is surveyed in subsection 1.2.4.

1.2.1 Rigid Manipulators

In describing a manipulator, a system of homogeneous transformations in the form of 4x4 matrices describing kinematics of lower pair mechanisms, developed by Denavit and Hartenberg [30], are used widely.

Uicker et al. [103] use these 4x4 matrices to describe transformations among manipulator frames. It seems to be rather unfortunate or illogical that they position the origin of each link coordinate system (frame) at the link distal end. The Q matrix that performs differentiation of a transformation matrix between neighboring links is introduced according to this convention.

It seems that until recently, the majority of papers followed the norm of positioning origins at the distal ends. The strongest influence for accepting this norm appears to be due to Paul [79].

Although many notations go under the Denavit-Hartenberg name, note that it is only important that any accepted norm is used consistently. Craig [27] uses a scheme where the origin of any link is positioned at its proximal end as seen from the universal or reference frame U.

The first frequently referenced manipulator control strategy, manipulator resolved rate control, is described by Whitney ([107] and [108]). Resolved rate control means that motion specifications in the form of initial and final positions and orientation of the end effector are given in base or hand Cartesian coordinates; the desired time needed for the entire motion is also given. The approach relies on the use of the Jacobian that can be seen as a transformation matrix from Cartesian space into joint space. The Jacobian is derived on the base of vector algebra which is claimed to be more effective than calculating it from kinematic relationships. The weak point is that this method requires the Jacobian inverse. Since the Jacobian does not have to be a square matrix a pseudoinverse of the Jacobian is often used.

Golla et al. [38] investigate control of rigid manipulators using a pole-placement

technique with linear state feedback. This method is applied to a linearized model of the manipulator. Two designs are proposed, first the centralized control, also called generalized rigid control (GRC), and then the decentralized control, named independent joint control (IJC). The manipulator model is further simplified by assumptions of no gravity and very slow motions (no Coriolis, centrifugal forces). Under these assumptions the dynamic equations of motions are in the form of double integrators and represent an unstable system. State feedback approach is used to stabilize this system. A proof is presented which shows that the linearized model can be stabilized and that the original fully nonlinear system is locally asymptotically stable. The GRC design is not unique since there are many state feedback matrices that yield the same closed-loop poles, but they differ in eigenvectors and consequently have different closed-loop responses, sensitivity to parameter variations and sensitivity to disturbances. The IJC cannot result in arbitrarily chosen closed-loop poles if the manipulator consists of more than two links. The open question even for the simplified model is the optimality of pole choice, i.e. type of the state feedback matrix. No nonlinearities of joint actuators have been considered.

Freund [34] proposes a fully nonlinear feedback controller that decouples the system dynamics. This method is significantly less complicated if applied in joint space than in cartesian space. It is essentially a pole-placement technique that is based on nonlinear system theory and decouples the coupled nonlinear systems. There are some problems associated with this approach because its accuracy depends on accurate parameter values, accurate on-line computations, accurate measurements of external forces and perfect rigidity. This method can perform relatively well even under high speed motions of manipulators.

Young [111] and [112] uses the theory of variable structure systems. The control algorithm is designed in such a way that it induces a so-called "sliding mode" in which the controlled system follows state space trajectory in the form of switching lines. The switching lines are not functions of manipulator parameters. Therefore the design parameters of switching lines define dynamics of the sliding mode. When in the sliding mode, the control law provides a compensation that eliminates nonlinear dynamic interactions of the manipulator joints. The sliding mode is always stable and the designer can choose a speed of convergence of the sliding mode to the origin of state space. Even if manipulator parameters are not known precisely, this design can be used knowing the bounds where the parameters exist. Ideal sliding mode (model parameters known precisely) generates a control signal that is switched at infinite frequency. If a nonideal sliding mode is induced then the control signal has a form

of a pulse-width/pulse-amplitude modulated signal. The required control signal has not been investigated from the point of view of a realistic joint actuator. This is important since the resulting control signal has the form of "chattering" which could cause damage to the actuator. Even in nonideal sliding mode, it may happen that dynamics of actuators cannot guarantee the required amplitudes and frequencies of the control signal - this problem deserves further investigation.

Luh et al. [66] propose resolved acceleration control of rigid manipulators. This method is an indirect extension of "resolved rate" and "computed torque" [27] control algorithms. The control torques generated by this algorithm guarantee that position and orientation errors go asymptotically to zero at each point of the preplanned trajectory. This method requires measurements of joint positions and velocities. The accelerations are obtained via differentiation of the Jacobian. Then, the recursive Newton-Euler method is used to obtain torques resulting in asymptotic convergence of both errors to zero. The rate of the asymptotic convergence is defined by two constants that can be specified by a designer. It has been pointed out that this algorithm requires substantial computational power and that problems with singularities of the Jacobian can occur.

Luh et al. [65] have also presented an extremely efficient algorithm for the solution of the "inverse dynamic problem", i.e. given desired joint positions, velocities and accelerations, the torques can be computed that, if applied in the "direct dynamic problem", would result in the desired joint positions, velocities and accelerations. This method uses recursive expressions for outward propagation of translational and rotational speeds and accelerations. It also calculates recursively the inward propagation of forces and moments. The entire algorithm is based on the Newton-Euler approach to equations of motions of rigid bodies. Another advantage of this algorithm is that it is general and requires only a knowledge of manipulator geometry and inertias of each link expressed with respect to their own mass centers. Inclusion of viscous damping is straightforward in this method. It is this algorithm that makes many other algorithms feasible in real-time. It can be also used for obtaining analytical expressions of manipulator dynamics.

Luh [64] has proposed a method for designing a controller for each joint if each actuator is controlled independently. This leads to the conventional design of joint controllers with requirements that dynamics of each link is critically damped using feedforward compensation and a conventional feedback controller. The design of this controller is done under the assumptions that only one link is moving and the remaining links are locked. This method when extended to a more natural situation, when

all links are moving at the same time, results in additional complicated feedforward compensation (inertia coupling, etc.). For a multi-link manipulator the computational requirements are quite significant (especially for feedforward compensations) and therefore simplification and parallel processing are suggested.

Murray and Newman [74] as well as Toogood [101] have proposed methods for automatic symbolic derivation of equations of motions. This approach is very convenient because the symbolic equations of motion can be programmed in an efficient way for real-time applications. Moreover, this approach provides easy and good insight into manipulator dynamics.

1.2.2 Manipulator Design

Although it may seem strange, manipulator design has not received much attention compared to other fields of robotics. Cost and needed team cooperation may be the reason why design of industrial manipulators is done mostly by robot manufacturing companies. Another reason could be that the first design and physical construction usually do not meet the desired specifications and such research is therefore rather expensive for the academic environment. Lesser recognition awarded to the design of a manipulator, as compared to that given to development and implementation of a successful control algorithm, may be an additional factor.

The majority of the manipulator design research has been done for manipulators using either direct drive (DD) actuators or actuators with harmonic drives (HD). The majority of research in the area of rigid DD manipulators has been done or supervised by H.Asada and T.Kanade, working on separate or joint projects.

Asada and Kanade [9] address some issues associated with the design of DD manipulators. They point out that the advantages of DD manipulators are no nonlinear effects due to transmission mechanisms (i.e. zero backlash and almost zero compliance at joints), small Coulomb friction, possibility of relatively very accurate control of positions and torques, smaller number of mechanical parts, etc. The essential problem with DD manipulators is the fact that a motor driving a certain joint becomes a load for the preceding joint motor. This may lead to an impractical design requiring very large motors to drive a heavy arm. It is recommended that rare-earth Samarium-Cobalt (Sa-Co) permanent magnet motors should be used because they have the highest available ratio of peak torque to motor weight.

Since it is difficult to know the dynamic characteristics of a manipulator at the design stage, Asada [4] uses a generalized inertia ellipsoid (GIE) to obtain an

insight into the manipulator dynamics. The GIE is obtained from the expression for the kinetic energy of the entire manipulator, and the resulting symmetric positive definite matrix is analyzed with respect to its eigenvectors. The eigenvectors are axes of GIE. The Lagrange approach is also applied, which together with GIE yields geometrically visible dynamic characteristics in the form of shape changes of GIE of the manipulator being designed.

Asada and Yamamoto [5] explain advantages of DD motors and use special high torque brushless motors for their MIT DD robot. They also point out problems associated with DD motors used. These motors need special servo amplifiers that compensate for delays in the dynamic response due to large windings (large time constants), i.e. inductance cannot be neglected. The torque exerted is not linear and contains a ripple due to pulsating current into windings which is related to the number of magnet poles used. This ripple in torque is about 5% of the exerted torque and the feedback current is not a reliable measurement of this ripple. To eliminate this torque ripple, a torque sensor consisting of an elastic spring coupler with strain gages is used. The torque measured by this sensor is proportional to joint acceleration, the joint position is measured by a synchro resolver, and the joint velocity is measured by a tachometer. These signals provide the necessary information to compensate the nonlinear effects including the ripple in torque.

Asada and Ro [6] show that force-speed characteristics can be improved either by DD motors with higher torques (resulting in bulkiness of actuators) or by modification of the link mechanism. This improvement is demonstrated on a special five-bar link mechanism. Their paper does not address the choice of a DD actuator that would satisfy the dynamic requirements, i.e. speed and acceleration along a preplanned trajectory.

Asada and Youcef-Toumi [7] as well as Youcef-Toumi and Asada [110] propose a strategy for manipulator design that can yield a manipulator with decoupled and/or configuration-invariant inertia tensors. Such a manipulator design drastically simplifies a design of a controller because, in extreme case, the manipulator can be treated as decoupled linear time invariant system. An example of this approach has been shown by Pak and Turner [77]. Unfortunately, decoupling can be obtained only for two-link manipulators and any payload which was not considered in the design phase will destroy decoupling and position invariance.

Schmitz et al. [85] describe a CMU (Carnegie-Mellon University) DD II manipulator. They conclude that brushless Samarium-Cobalt motors are the most suitable for DD applications. They also emphasize the necessity for extremely high resolution

sensors for measuring angular position and velocity. They use a two-speed pancake resolver with a current feedback on their servoamplifier.

Khosla and Kanade [51] use the CMU DD II described above and report an achieved sampling time period of 1 msec while using a computed torque control algorithm. They use a tightly coupled multiprocessor system and a special Marinco processor added for fast floating point operations. This manipulator is designed mostly for motions in the horizontal plane and avoids excessive torques which will otherwise be needed to overcome gravitational effects.

Takase [96] describes a 6-link rigid manipulator with DD frameless Sa-Co D.C. motors. The third link is counterbalanced since it moves in the vertical plane. High resolution optical encoders are used for shaft position measurements.

Whitney [109] provides a list of considerations that should be thought of before any design begins. Kiedrzynski [52] concludes that in case of rigid manipulators it is more convenient to use aluminum than steel. If link weight is defined as a criterion, then steel links are more susceptible to buckling than aluminum links of the same mass.

Beck and McCloy [13] compare the power requirements of several different configurations of two-link manipulators. Their analysis concludes that 2R (both joint rotational) manipulators are by far the least efficient, however their dexterity makes them popular.

Takeshita et al. [97] develop a rigid DD manipulator using high performance Sa-Co A.C. (brushless) motors and use the built-in resolver for position measurements. Their robot is of SCARA (selective compliance assembly robot arm) type, which does not have to deal with gravity effects. Maximum velocity is 7 m/s and maximum acceleration is 4 g.

Canudas et al. [21] indirectly motivate a choice of DD manipulators. Their paper deals with adaptive compensation of nonlinear friction effects of a one-link manipulator. Their results show severe effects of friction on the dynamic response if no adaptation is used. Therefore one can expect that friction effects would profoundly influence manipulator dynamics in case of a multi-link manipulator.

Kubo et al. [57] analyze the effects of friction on a D.C. motor with harmonic drive (HD) transmission where the nonlinear friction effects are mostly due to HD. They show that stability and dynamic performance of such a servosystem with HD depend strongly on the magnitude and class of input signals. They propose a nonlinear controller with Coulomb friction compensation to improve the performance and stability of their servosystem.

Potkonjak and Jaksic [80] provide a procedure for choice of D.C. motors with gear trains. This method takes into account the complete dynamics of the controlled manipulator mechanisms. A motor is evaluated for sufficient power, acceleration, and jerk power. One important conclusion is that to consider any motor as only a first order system is incorrect.

Sweet and Good [95] introduce important and realistic considerations related to present manipulators. Standard equations of motion for rigid manipulators typically do not include a very important part of each manipulator, namely the dynamic interactions with the drive system (actuators). Very frequently, torque actuators are modeled as pure torque sources or as a first order lag. If a HD is used, it introduces mechanical compliance and significant resonance/antiresonance phenomenon into the dynamics of the manipulator. If any controller is intended to decouple the manipulator dynamics, it must take into consideration the real dynamics of the actuators. Further, nonlinear effects of actuators like friction and saturation must be modeled as well. They also pointed out that if high-gear ratios are employed then torques resulting from cross-coupling as reflected back to the motors are minimal. It is also pointed out that because the majority of algorithms are going to be implemented via digital computers, effects of finite sampling rate, amplitude quantization and uncertain information from sensors should be investigated.

Graettinger and Krogh [39] present a measure of manipulator performance in the form of acceleration radius. The acceleration radius is defined as maximum norm of end point cartesian acceleration that can be achieved at any point of the manipulator workspace which leads to a numerical optimization problem. The acceleration radius can be used for determination of actuator torques if a minimum acceleration radius is given. It is not clear how to obtain this minimum acceleration radius objectively if only a limited subset of manipulators tasks are known in advance, i.e. it can lead to overdetermination of actuators.

1.2.3 Adaptive Control of Manipulators

Adaptive control strategies have been proposed for control of mechanical manipulators to achieve insensitivity to parameter uncertainties in manipulator and actuator dynamics as well as to unknown load variations.

Dubowsky and DesForges [32] present the first adaptive control algorithm for rigid manipulators. They use model reference adaptive control (MRAC) strategy, where the reference model is a decoupled system of linear-time-invariant (LTI) second

order systems. Their approach is based on a technique by Donaldson and Lendes [31] which guarantees good results under the assumption that the rate of change of manipulator dynamics is much slower than adaptive parameter updates. Their simulations show that consistent manipulator performance is maintained over a wide range of system motions and payloads.

Durrant-Whyte [33] develops MRAC strategy based on Lyapunov stability criterion and therefore stability is guaranteed for any input. His model of the rigid manipulator includes such physical considerations as motor saturation, static friction, unknown loads and uncertain plant parameters. Again, the reference model is a set of decoupled second order LTI systems. The adjustable system is introduced via state feedback gains. To obtain a closed-form for updates of state feedback gains, the assumption is made that the rate of evaluation of manipulator dynamics is small compared to the rate of the feedback rate updates. No a priori knowledge of manipulator dynamics is required.

Koditschek [53], [54] presents a stable adaptive proportional-derivative (PD) state feedback law for rigid manipulators. This law updates changing parameters of the manipulator dynamics. His method relies on natural unforced motions of manipulators, i.e. no reference model is needed.

Craig et al. [26] and Craig [28] design a globally asymptotically stable adaptive control law for rigid manipulators based on technique by Parks [78]. To avoid possible large variations of estimated parameters of manipulator dynamics, parameter bounds are set up to enable resetting whenever estimated parameters are out of their bounds.

Kathab [50] uses MRAC together with Popov's hyperstability theorem and properties of positive realness to obtain state feedback adaptive control scheme for manipulators. This model can be added to PD model based (computed torque) controller and it is claimed that good trajectory tracking is achieved while performing on-line estimation of uncertain parameters.

Horowitz and Tomizuka [46] use also MRAC with Popov's hyperstability theorem. Their system contains two feedback loops. The inner loop is the MRAC controller and the outer loop is a proportional-integral-derivative (PID) controller. Effects of actuator dynamics and nonlinearities are not considered.

Tomizuka [100] summarizes research in adaptive control of rigid manipulators and concludes that many proposed algorithms were tested by simulations only. A successful adaptive algorithm should perform well in real experiments while taking care of effects of friction forces, actuator dynamics, adaptation speed, measurements, etc. His results, from two experiments, show that a properly designed adaptive control

strategy has more consistent performance if compared to any nonadaptive control law. It is pointed out that adaptive control is particularly attractive for high performance direct drive arms.

Atkeson et al. [11] present a method for estimation of inertial parameters of manipulators and their payloads. This method has been experimentally verified and it has been shown that consistently better results are obtained from a direct drive manipulator with large torques. Analysis shows that there are some inertial parameters that cannot be estimated because their influence on manipulator dynamics is very small. Therefore, parameters that can be estimated are sufficient for good control strategies. The discussed results are obtained from off-line identification, however they can be calculated on-line recursively if sufficient computational power is available.

Seraji [87] uses improved Laypunov-based MRAC method to enhance a non-adaptive control law consisting of feedforward and feedback controllers. The feedforward controller is chosen as inverse of the linearized manipulator model and the feedback controller is a PD controller. The proposed adaptive law updates parameters of both the feedforward and the feedback controllers. Additionally, an auxiliary signal is generated to improve closed-loop stability and to achieve faster adaptation. The reference model is again a set of decoupled second order LTI systems. The proposed adaptation law does not require knowledge of the complex model of the manipulator dynamics. Information obtained from actual and reference trajectory is sufficient for a reasonable control performance.

Meldrum and Balas [70], [71] develop a MRAC controller for a flexible manipulator. The reference model is obtained by considering only a finite number of modes describing vibrations of the flexible manipulator. The closed-loop adaptive control law is stable if the estimated open-loop is output stabilizable and if the closed-loop adaptive control law satisfies the positive realness condition (Nyquist plot in right half plane). Simulation results are presented to show feasibility of MRAC for flexible manipulators. The simulation results imply that the positive realness condition is a function of the location of actuators and sensors.

1.2.4 Experimental Research of Flexible Manipulators

Most of the experimental research on flexible manipulators has been done with one flexible link that can move only in a horizontal plane and exhibits transverse vibrations in the horizontal plane (other vibrations are prevented intentionally). This approach

enables one to neglect effects of gravity.

Cannon and Schmitz [20] have pointed out the principal difficulties connected with controls of noncolocated systems (i.e. systems where positions of sensors and actuators are different). They use a link with fundamental modal frequency 2 Hz and their system includes the first three modes. Optical sensing of the end-point position is used. Design of a LQG compensator is demonstrated with a sampling frequency of 50 Hz. Discussion of plant/model mismatch shows significant sensitivity of the controller (i.e. stability problems). Experimental results demonstrate that the tip of the link moves only after the bending-type wave propagates the entire length of the link (the wave is initiated by move of the joint actuator). The link is 1 m long and a typical step change in set-point is 0.1 m.

Both Maples [67] and Schmitz [86] use the same very flexible manipulator built at Stanford and a model by Gevarter [36] that describes a system with linear distributed flexibility as an infinite number of parallel second order systems. While Schmitz concentrates his effort on precise closed-loop position control, Maples extends this work into force control of the above mentioned one-link flexible manipulator. Both approaches are essentially based on a linearized model that includes the first three modes, i.e. appropriate types of transfer functions were specified and the parameters used are determined experimentally. The open-loop transfer functions for noncolocated sensors are non-minimum phase type transfer functions with one unstable zero. The closed-loop controllers and estimators are of the discrete LQG type.

The conclusions of the above works suggest that use of a light accelerometer at the tip of the link would be beneficial. Even more important would be an on-line identification and a complete adaptive control scheme. Further extension of their work should be a high-bandwidth feedback compensator that handles large angular steps and allows for high-speed motions. The ultimate goal should be a high-bandwidth 6 degree of freedom (DOF) manipulator with three dimensional position sensors, together with higher level control that should incorporate servocontrol of loops with trajectory planning, sensor monitoring, failure detection and automated recovery. On-line identification of modal (resonant) frequencies would make control of flexible manipulators more robust because modes are functions of payloads and positions.

Hastings and Book ([41], [43]) also study a one-link flexible manipulator moving in a horizontal plane. They use the Lagrangian approach with consideration of the first three modes. The resulting equations are linearized under assumptions of small motions. The LQ regulator approach is used for controller design. Since direct

measurement of modes is not feasible, only quantities that are linear combinations of modes or their derivatives (displacements, etc.) can be obtained. For this reason, strain gages in full bridge (to eliminate effects of undesired torsional, axial, and transverse strains on measurements) are used. The purpose of their experiments is to verify agreement between theoretical and practical results. Although several boundary conditions are considered (clamped-free, clamped-mass, pinned-free, pinned-mass, etc.), the experimental and theoretical results are close only in clamped-mass boundary case. The first three modal frequencies are 2, 13.9, and 42.8 Hz and the maximal sampling frequency is 100 Hz.

Gebler [35] reports experimental results with a two-link flexible manipulator with both links moving in one vertical plane. Under the assumption of small vibrations, equations of motions (using D'Alambert principle and Lagrange approach) are linearized around a nominal trajectory. Joint actuators are modeled as linear lumped systems. Feedforward compensation is shown to significantly improve the dynamic response and trajectory tracking. Measurements of modal frequencies of the entire manipulator in different positions reveal changes in modal frequencies in the range between 18% and 33%. Stretched positions result in lower modal frequencies while folded positions are characterized by higher modal frequencies.

Chalhoub and Ulsoy [22] have performed experiments and simulations with a two-link spherical manipulator in which the second link is flexible. Both transverse vibrations are considered and a mass-point is used as a payload. They derive two controllers. The first one is "rigid body controller" that does not take into account the flexibility of the second link and the other is a "rigid and flexible motion controller". Assumed modes method for the first two modes and Lagrange approach are employed for the development of equations of motions. Based on a linearized model of the developed equations, the two controllers (both in the form of integral and state feedback) are designed. This configuration of the controllers is particularly useful because it enables decoupling of the linearized system and independent design for each axis. The controller designs have been done while considering none, first, and both modes to investigate effects of observation and control spillover. The model is described by a linear time-invariant MIMO system and the pole placement technique is used. Better transient response is achieved if state variables retain their physical meaning (no transformation to obtain equivalent SISO system). Poles are chosen in such a way that axes have critical damping and modes have increased damping. They show that some structural damping is essential for simulations, otherwise, in certain situations, unstable response is obtained. No modeling of actuator nonlinearities is

evident (although motors with gears are used) and that could be the reason for the unstable situations observed by Kubo et al. [57]. The considered modal frequencies are 6 Hz and 43 Hz. An important conclusion is that the joint actuators must have adequate bandwidth to generate high-frequency torque changes needed to damp out the flexible motion.

Lambert [59] has performed experiments controlling a one-link flexible manipulator in the horizontal plane using the generalized predictive control (GPC) method developed by Clarke et al. [25], with an on-line recursive least square (RLS) identification scheme. The GPC algorithm is, among other reasons, quite suitable because it can handle non-minimum phase systems (structures with distributed flexibility are described by non-minimum phase transfer functions if linearized). To cope with disturbances and to preserve stability, certain filters (T-filter, etc.) have to be used to limit frequency range of disturbances. The first two modal frequencies are 15 Hz and 50 Hz, however only the first modal frequency is included in the system to be controlled because the sampling rate is 60 Hz. To eliminate nonlinear effects of the gear box, a direct drive motor is used with high resolution optical encoders (5000 pulses per rotation).

Kotnik et al. [55] compared experimentally several controllers to damp out the flexible link vibrations in the horizontal plane after a step command in the motor shaft position was applied. They showed that the controllers, which use the flexible tip acceleration feedback, perform better than controllers based on observer design or the endpoint position feedback.

Alberts et al. [1] carried out experiments with combined active and passive damping of the flexible link vibrations. The active damping design uses an inverse dynamics approach which is, in turn, based on the solution of finite element equations requiring knowledge of the manipulator model. The passive damping requires use of special viscoelastic materials in the link material and reduces oscillations resulting from plant/model mismatch.

Identification and control experiments with a planar 2 link flexible manipulator are described by Yourkovich et al. [113]. The investigated controllers are based on the simplified assumption of independent joint control. The vibrations of each link are sensed by accelerometers and the resulting control is found satisfactory as long as the identified parameter variations were small.

1.3 Contributions of this Thesis

The main contributions of this thesis are:

- The design and construction of the 2 DOF flexible manipulator driven by direct-drive motors and the design and building of all subsystems so that a test-bed for future research in the area of the flexible manipulators will be available in the Electrical Engineering Department. To the best knowledge of the author, the existence of such a test-bed has not been reported in the literature.
- Development of an adaptive MIMO GPC algorithm and its implementation.
- Achieving reasonable success in experiments in controlling the 2 DOF flexible manipulator with two control strategies: 1) Adaptive MIMO GPC; and 2) Multivariable PID control law.
- Implementation of the adaptive MIMO GPC in a parallel processing environment, using transputers.

1.4 Organization of the Thesis

This chapter explains the justification of this project and its scope. Additionally, it provides a limited literature survey of related work in this area. A full scale survey is beyond the scope of this thesis since the number of related published works is very large. In addition, the organization and the contributions of this thesis are discussed.

Chapter 2 deals with the mechanical design of the 2 DOF flexible manipulator. Typical trajectories considered for the manipulator design are presented and used for calculations of needed torques using the Newton-Euler dynamic algorithm. The choice of the direct drive motors and the flexible link are also discussed.

The design and construction of the hardware and some software subsystems are discussed in chapter 3. These include, for example, interface subsystems, power distribution subsystem, software drivers etc.

Chapter 4 presents the first tests of the 2 DOF flexible manipulator. The friction effects and their compensation are also described. The frequencies of the flexible link vibrations under different conditions are measured; and a methodology to use the double integrator interface is developed in this chapter.

Chapter 5 describes the development and some implementation issues for the adaptive MIMO GPC which was chosen from a large number of adaptive control

algorithms to control the 2 DOF flexible manipulator. The implementation of on-line MIMO recursive prediction error method (RPEM) identification method is discussed.

Simulation results of the MIMO GPC controlling output coupled and output decoupled nonsquare systems with smaller number of inputs than outputs are presented in chapter 6. The effects of some tuning knobs are also demonstrated and discussed here.

Experimental results of controlling the 2 DOF flexible manipulator are presented in chapter 7. Two algorithms, the adaptive MIMO GPC and the multivariable PID algorithms, are tested and their results are discussed.

Chapter 8 discusses issues related to the implementation of the adaptive MIMO GPC in the transputer parallel processing environment. Some implications for software development in the parallel processing are presented.

Finally, chapter 9 presents conclusions and recommendation for future research.

Chapter 2

Manipulator Design

2.1 Introduction

The process of manipulator design is iterative. It is not an orderly unidirectional process from specifications through a series of design decisions to a device. It must accommodate different (sometimes contradictory) constraints which appear during the design process. Additionally problems occur when the values of certain parameters turn out to be different from assumed values. This information becomes available in some cases only after the design is completed and the components are acquired. In such cases, the design will must be repeated. It has been suggested by some researchers that each manipulator prototype must frequently be designed twice.

Model development and design of the robotics manipulator are interlinked. The development of an analytical model of a manipulator requires a good knowledge of mechanical engineering principles. Simplifying assumptions can be used if the purpose of the model is to describe dominant behavior. However, when such a model is used in an actual control experiment, numerical values of many parameters and variables must be known. Inaccurate parameters and the neglected dynamics play important roles in the performance of the controlled manipulator.

Similarly, the design of a robotics manipulator is a process during which many assumptions are made. Only when the system is built and the first tests start, deficiencies appear. During the design process, a designer must accommodate unexpected modifications through the flexibility of the design. If something has to be changed then it should result in the minimum modifications of the affected subsystems. The manipulator designer cannot test components before they are purchased. Actual values of many of the parameters are obtained only after the manipulator is built.

In this chapter the various factors, that were taken into account in the design of the flexible manipulator, are described. This chapter is organized as follows. In section 2.2 the manipulator configuration, its workspace and intended task are described. Section 2.3 describes a realistic velocity profile. Based on this profile, the anticipated maximum velocities and accelerations are calculated. These values will be used in section 2.5. In section 2.4 kinematics and coordinate frames for the manipulator are described. Section 2.5 deals with the inverse dynamics calculations for the manipulator. The purpose is to determine the necessary torques that the actuators (motors) will have to develop to carry out the prescribed task. In section 2.6, the choice of motors is discussed. The last section 2.7 focuses on the choice of the size and shape of the flexible link.

Although the objective is to design a manipulator with one flexible link, the calculations in sections 2.4 and 2.5 are carried out assuming that both links are rigid and a torque-speed characteristics are obtained. However, when choosing motors (section 2.6) sufficient allowance is made to handle flexibility effects.

2.2 Configuration and Task Specification

To develop a test-bed for the control of two dimensional vibrations, the robotic manipulator must have two degrees of freedom (2 DOF). The chosen 2R (two revolute joints) configuration is shown in Figure 2.1. This manipulator has two motors; the base and shoulder motor whose positions are described by the angles θ_1 and θ_2 respectively.

The manipulator workspace is shown in Figure 2.2, in which $\theta_1 \in <0^\circ, 360^\circ>$, i.e. one full rotation, and $\theta_2 \in <-160^\circ, 160^\circ>$. Angle $\theta_2 = 0^\circ$ means that the second link is in vertical position. Lengths of both links are assumed to be 1 m each.

Although many definitions for the manipulator required performance can be used, the following definition in the form of the desired manipulator task is chosen due to its generality. *The time for motion from any initial point to any final point in the manipulator workspace should not take more than 1 sec.*

2.3 Trajectory Considerations

To determine the desired torque-speed motor characteristics, it is necessary to identify :

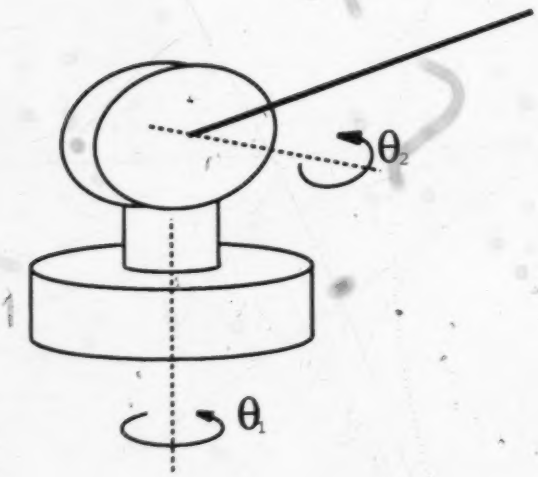


Figure 2.1: Manipulator configuration.

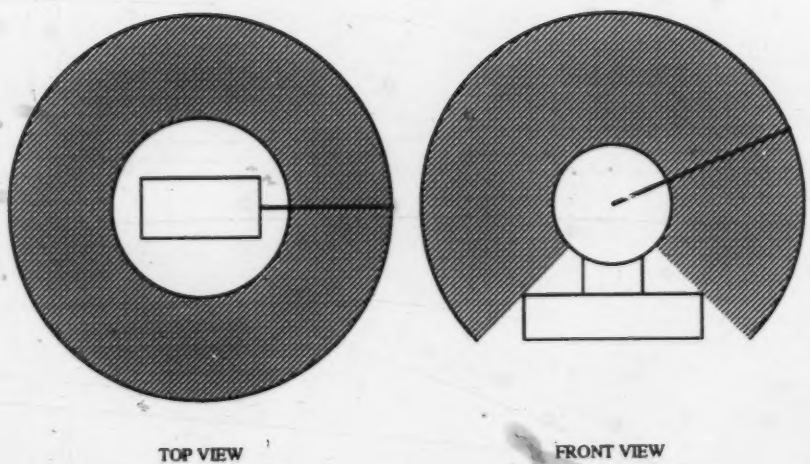


Figure 2.2: Manipulator workspace.

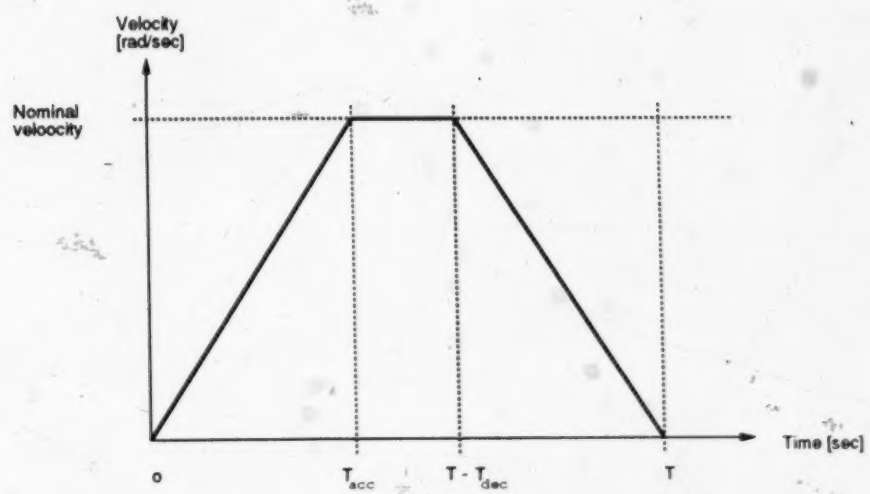


Figure 2.3: Piece-wise linear velocity profile.

- the worst scenario trajectory in the form of positions, velocities, and accelerations
- manipulator inertias
- torques due to friction, payload, vibrations, and gravitational effects.

If the velocity profile of a joint is known, it is possible to calculate positions and accelerations associated with the joint trajectory. The constant velocity of the joint when all the torques acting on the joint are in equilibrium is called the nominal velocity and denoted by ω_{nom}.

A classic textbook velocity profile is shown in Figure 2.3. This type of the velocity profile has at least two shortcomings. The first is that the acceleration and deceleration are not continuous, implying non-zero initial value of the jerk and sudden changes in torque which are undesirable for control of vibrations. The second shortcoming is that the delivered torque must be available at nominal velocity, which is an unrealistic demand on the actuator.

Therefore a quasi trapezoidal velocity profile as shown in Figure 2.4 [80] is used to obtain expressions for needed kinematic quantities. As will be seen in the sequel, this profile guarantees that all kinematic quantities will be continuous. This profile consists of three sections with four boundaries at times 0, T_{acc}, $T - T_{dec}$, and T.

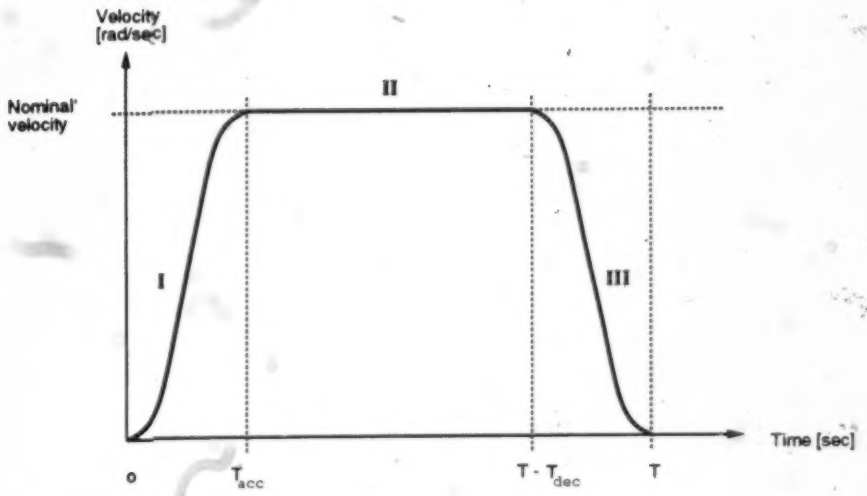


Figure 2.4: Quasi-trapezoidal velocity profile.

To develop unique expressions for the kinematic quantities it is necessary to impose constraints in the form of zero boundary jerk, and matching velocities and accelerations at the boundaries between particular trajectory sections. Sections I and III are assumed to be sinusoidal so that analytical formulas could be obtained. The conditions can be stated as follows :

- To avoid jerk effect, the first derivative of the accelerations ϵ_I and ϵ_{III} must be continuous. This means, that

$$\dot{\epsilon}_I(0) = \dot{\epsilon}_I(T_{acc}) = \dot{\epsilon}_{III}(T - T_{dec}) = \dot{\epsilon}_{III}(T) = 0 \tag{2.1}$$

- Acceleration and deceleration are zero at path section boundaries.

$$\epsilon_I(0) = \epsilon_I(T_{acc}) = \epsilon_{III}(T - T_{dec}) = \epsilon_{III}(T) = 0 \tag{2.2}$$

- Angular velocity ω should reach its nominal value ω_{nom} at $t = T_{acc}$ and its initial and final values should be zero.

$$\omega_I(T_{acc}) = \omega_{nom} \tag{2.3}$$

$$\omega_I(0) = \omega_{III}(T) = 0 \tag{2.4}$$

- The initial conditions for the angular position are zero,

$$\theta(0) = 0 \tag{2.5}$$

The expressions for jerk, acceleration, velocity, and position for the three sections are given below.

Section I for $t \in < 0, T_{acc} >$

$$\dot{\epsilon}_I(t) = \frac{2\pi\omega_{nom}}{T_{acc}^2}\cos(\frac{2\pi}{T_{acc}}t + \frac{3\pi}{2}) \tag{2.6}$$

$$\epsilon_I(t) = \frac{\omega_{nom}}{T_{acc}}(\sin(\frac{2\pi}{T_{acc}}t + \frac{3\pi}{2}) + 1) \tag{2.7}$$

$$\omega_I(t) = \frac{\omega_{nom}}{T_{acc}}(t - \frac{T_{acc}}{2\pi}\cos(\frac{2\pi}{T_{acc}}t + \frac{3\pi}{2})) \tag{2.8}$$

$$\theta_I(t) = \frac{\omega_{nom}T_{acc}}{(2\pi)^2}(-1 - \sin(\frac{2\pi}{T_{acc}}t + \frac{3\pi}{2})) + \frac{\omega_{nom}t^2}{2T_{acc}} \tag{2.9}$$

Section II for $t \in < T_{acc}, T - T_{dec} >$

$$\dot{\epsilon}_{II}(t) = 0 \tag{2.10}$$

$$\epsilon_{II}(t) = 0 \tag{2.11}$$

$$\omega_{II}(t) = \omega_{nom} \tag{2.12}$$

$$\theta_{II}(t) = \theta(T_{acc}) + \omega_{nom}(t - T_{acc}) = \frac{1}{2}\omega_{nom}T_{acc} + \omega_{nom}(t - T_{acc}) \tag{2.13}$$

Section III for $t \in < T - T_{dec}, T >$

$$\dot{\epsilon}_{III}(t) = \frac{2\pi\omega_{nom}}{T_{dec}^2}\cos(\frac{2\pi}{T_{dec}}(t - T + T_{dec}) + \frac{\pi}{2}) \tag{2.14}$$

$$\epsilon_{III}(t) = \frac{\omega_{nom}}{T_{dec}}(\sin(\frac{2\pi}{T_{dec}}(t - T + T_{dec}) + \frac{\pi}{2}) - 1) \tag{2.15}$$

$$\omega_{III}(t) = \omega_{nom} - \frac{\omega_{nom}}{T_{dec}}(\frac{T_{dec}}{2\pi}\cos(\frac{2\pi}{T_{dec}}(t - T + T_{dec}) + \frac{\pi}{2}) + (t - T + T_{dec})) \tag{2.16}$$

$$\begin{aligned}\theta_{III}(t) \;=\; & \theta_{|(T-T_{dec})} + \omega_{nom}(t - T + T_{dec}) \\ & -\frac{\omega_{nom}}{T_{dec}}(\frac{T_{dec}^2}{(2\pi)^2}(\sin(\frac{2\pi}{T_{dec}}(t - T + T_{dec}) + \frac{\pi}{2}) - 1) + \frac{1}{2}(t - T + T_{dec})^2)\end{aligned} \tag{2.17}$$

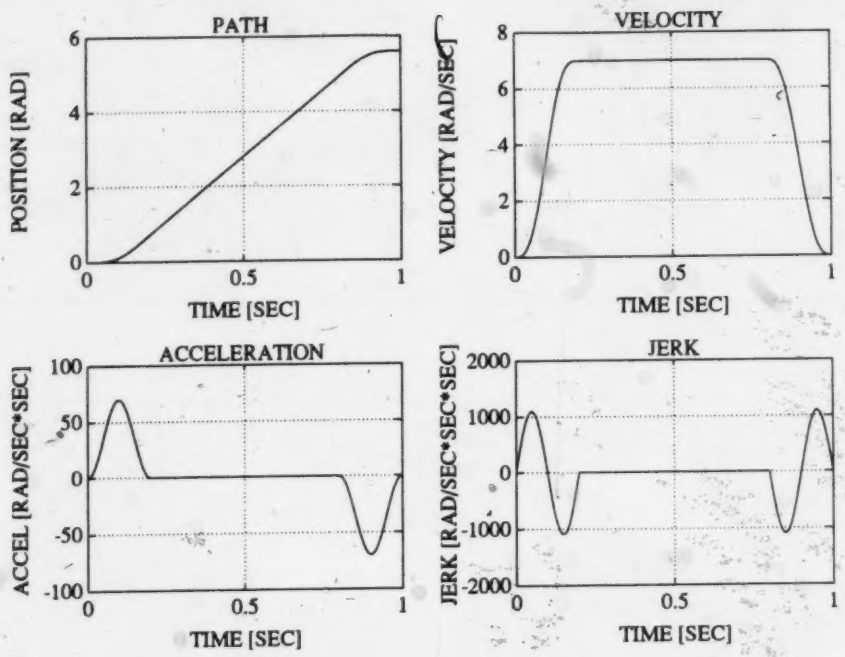


Figure 2.5: Kinematic quantities with zero boundary jerk.

Since

$$\theta_{|(T-T_{dec})} = \frac{1}{2}\omega_{nom}T_{acc} + \omega_{nom}(T - T_{dec} - T_{acc}) \tag{2.18}$$

we can calculate the final angular position $\theta_{III}(T)$

$$\theta_{III}(T) = \frac{1}{2}\omega_{nom}(2T - T_{dec} - T_{acc}) \tag{2.19}$$

The above expression is convenient for calculation of ω_{nom} if we know $\theta_{III}(T)$, T, T_{acc}, and T_{dec}

$$\omega_{nom} = \frac{2\theta_{III}(T)}{2T - T_{acc} - T_{dec}} \tag{2.20}$$

The nominal angular velocity can be obtained using the above equation as a result of the initial choices for the desired path. The shape of all kinematic quantities is illustrated in Figure 2.5.

Since the maximum acceleration occurs at $t = T_{acc}/2$ and minimum acceleration (maximum deceleration) at $t = T - T_{dec}/2$, it desirable to know their values and corresponding velocities.

$$\epsilon_{I_{MAX}}(T_{acc}/2) = \frac{\omega_{nom}}{T_{acc}/2} = \frac{4\theta_{III}(T)}{T_{acc}(2T - T_{acc} - T_{dec})} \tag{2.21}$$

$$\omega_I(T_{acc}/2) = \frac{\omega_{nom}}{2} \tag{2.22}$$

$$\epsilon_{III_{MIN}}(T - T_{dec}/2) = -\frac{\omega_{nom}}{T_{dec}/2} = -\frac{4\theta_{III}(T)}{T_{dec}(2T - T_{acc} - T_{dec})} \tag{2.23}$$

$$\omega_{III}(T - T_{dec}/2) = \frac{\omega_{nom}}{2} \tag{2.24}$$

The derivation is repeated allowing non-zero boundary jerk. The resulting equations for all three path sections follow.

Section I for $t \in < 0, T_{acc} >$

$$\epsilon_I(t) = \frac{\omega_{nom}\pi}{2T_{acc}} \sin(\frac{\pi}{T_{acc}}t) \tag{2.25}$$

$$\omega_I(t) = \frac{\omega_{nom}}{2}(1 - \cos(\frac{\pi}{T_{acc}}t)) \tag{2.26}$$

$$\theta_I(t) = \frac{\omega_{nom}}{2}(t - \frac{T_{acc}}{\pi} \sin(\frac{\pi}{T_{acc}}t)) \tag{2.27}$$

The **section II** for $t \in < T_{acc}, T - T_{dec} >$ is described by an identical set of equations as in the zero boundary jerk case.

Section III for $t \in < T - T_{dec}, T >$

$$\epsilon_{III}(t) = -\frac{\omega_{nom}\pi}{2T_{dec}} \sin(\frac{\pi}{T_{dec}}(t - T + T_{dec})) \tag{2.28}$$

$$\omega_{III}(t) = \frac{\omega_{nom}}{2}(1 + \cos(\frac{\pi}{T_{dec}}(t - T + T_{dec}))) \tag{2.29}$$

$$\theta_{III}(t) = \theta_{|(T-T_{dec})} + \frac{\omega_{nom}}{2}(t - T + T_{dec} + \frac{T_{dec}}{\pi} \sin(\frac{\pi}{T_{dec}}(t - T + T_{dec}))) \tag{2.30}$$

We also have

$$\epsilon_{I_{MAX}}(T_{acc}/2) = \frac{\omega_{nom}\pi}{2T_{acc}} = \frac{\pi\theta_{III}(T)}{T_{acc}(2T - T_{acc} - T_{dec})} \tag{2.31}$$

$$\omega_I(T_{acc}/2) = \frac{\omega_{nom}}{2} \tag{2.32}$$

$$\epsilon_{III_{MIN}}(T - T_{dec}/2) = -\frac{\omega_{nom}\pi}{2T_{dec}} = -\frac{\pi\theta_{III}(T)}{T_{dec}(2T - T_{acc} - T_{dec})} \tag{2.33}$$

$$\omega_{III}(T - T_{dec}/2) = \frac{\omega_{nom}}{2} \tag{2.34}$$

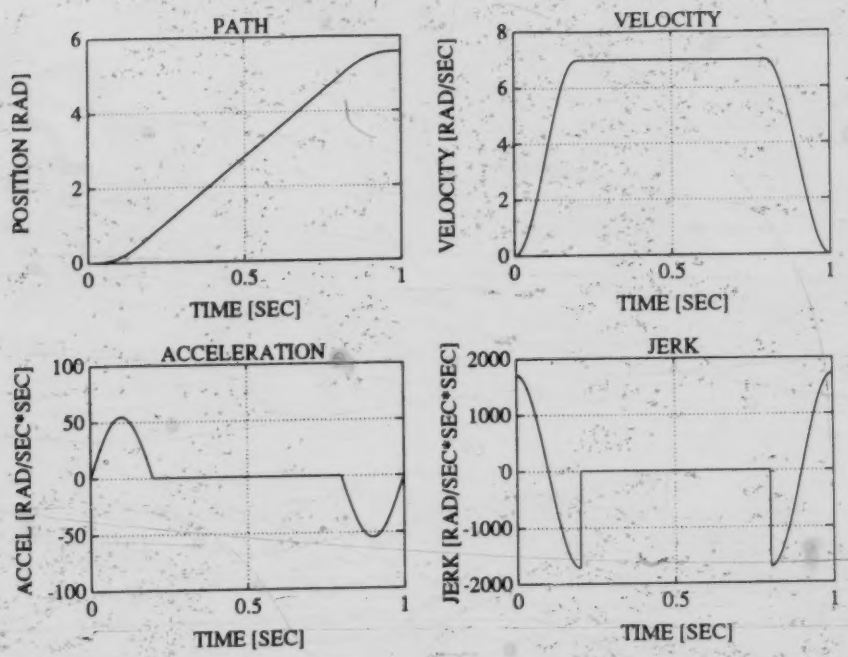


Figure 2.6: Kinematic quantities with nonzero boundary jerk.

The jerk, during the first and third path sections, is

$$\dot{\epsilon}_I(t) = \frac{\omega_{nom}\pi^2}{2T_{acc}^2}\cos(\frac{\pi}{T_{acc}}t), \qquad for \quad t \in < 0, T_{acc} > \tag{2.35}$$

$$\dot{\epsilon}_{III}(t) = \frac{\omega_{nom}\pi^2}{2T_{dec}^2}\cos(\frac{\pi}{T_{dec}}(t - T + T_{dec})), \qquad for \quad t \in \leq T - T_{dec}, T > \tag{2.36}$$

The typical shapes of all above developed kinematic quantities are shown in Figure 2.6.

Although both solutions yield the same nominal angular velocities for the same timing and distance requirements, the solution with non-zero boundary jerk results in smaller values of maximum and minimum accelerations. This is to be expected because in the case of non-zero jerk the acceleration can increase more rapidly at the beginning of the acceleration sections.

The ratio of maximum accelerations for the two methods is 1.273. Tables 2.1 and 2.2 provide information about nominal velocities and maximum accelerations for different timing requirements when the workspace is defined by $\theta_1 \in < 0°, 320° >$ and $\theta_2 \in < 0°, 180° >$. If the velocity profile shown in Figure 2.3 is compared, for example, for the base motor with $T_{acc} = 0.2$, the resulting constant acceleration is 34.9

	Nonzero Boundary Jerk				Zero Boundary Jerk			
$T_{acc} = T_{dec}$ [sec]	0.05	0.10	0.15	0.20	0.05	0.10	0.15	0.20
ω_{nom} [rad/s]	5.87	6.20	6.57	6.98	5.87	6.20	6.57	6.98
$\epsilon_{I_{max}}$ [rad/s^2]	184.69	97.47	68.80	54.83	235.16	124.11	87.61	69.81

Table 2.1: Velocity and acceleration for the base motor.

	Nonzero Boundary Jerk				Zero Boundary Jerk			
$T_{acc} = T_{dec}$ [sec]	0.05	0.10	0.15	0.20	0.05	0.10	0.15	0.20
ω_{nom} [rad/s]	3.30	3.49	3.69	3.93	3.30	3.49	3.69	3.93
$\epsilon_{I_{max}}$ [rad/s^2]	103.89	54.83	38.70	30.84	132.27	69.81	49.27	39.26

Table 2.2: Velocity and acceleration for the shoulder motor.

[rad/sec^2] which is significantly smaller than the corresponding value 54.86 [rad/sec^2] in the Table 2.1. However, the acceleration needed for the piece-wise linear velocity profile must be available over the entire velocity range which is extremely demanding for typical motors, because their torque-speed characteristics exhibit decreasing available torque with increasing velocity. The nominal velocities are identical because they depend, in both cases, only on T, T_{acc}, and T_{dec}.

The previously derived equations provide sufficient information to choose an appropriate actuator, only for 1 DOF manipulator when zero friction, no gravity, and constant inertia are assumed. This is an ideal situation in which the acceleration is proportional to torque. However, these assumptions are not valid when manipulators with more than one degree of freedom are considered and when coupling moments, friction, and gravitational effects must be taken into account.

The results of this section will be used in section 2.5 in the inverse dynamics calculations. Before this can be done it is necessary to describe the manipulator kinematics.

2.4 Manipulator Kinematics

Manipulator kinematics are described by link transformations which describe mapping between link frames. The link transformations are referred to as Denavit-Hartenberg notation [30]. Following Craig [27] the origin of i-th frame is assumed to be at the proximal joint of the i-th link.

Figure 2.7 shows the 2 DOF manipulator with a payload in the form of an additional link without any actuator. This non-actuated link has been selected to represent a payload represented by an inertia tensor and to allow for future extensions

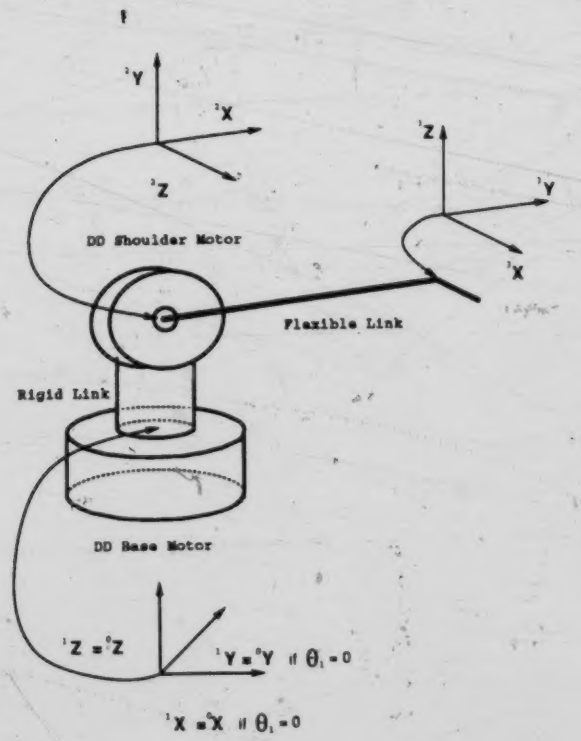


Figure 2.7: 2 DOF robotics manipulator.

to a 3 DOF flexible manipulator. All frames are drawn with joint angles equal to zero. The reference frame $\{0\}$ is placed such that it is identical with frame $\{1\}$ when $\theta_1 = 0$. The origin of the frame $\{2\}$ is placed at the intersection of axes of the base and shoulder motors. To derive the homogeneous transformation matrices ${}^{i-1}T_i$ between the consecutive frames $\{i-1\}$ and $\{i\}$, the method of successive elementary homogeneous transformations is used. This method consists of a sequence of rotations and translations which transform frame $\{0\}$ to frame $\{1\}$, frame $\{1\}$ to frame $\{2\}$, etc.

The homogeneous transformation ${}^0T_1$ describing relation between frame $\{0\}$ and frame $\{1\}$ is

$$ {}^0T_1 = ROT(z, \theta_1) \tag{2.37} $$

with its homogeneous transformation

$$ {}^0T_1 = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.38} $$

To obtain the next homogeneous transformation it is necessary to perform translation of $l_1$units along the z axis of frame $\{1\}$. This results in an intermediate frame with its origin at the desired origin of frame $\{2\}$. The intermediate frame is rotated by 90^o around its x axis so that the rotated z axis is aligned with the axis of the shoulder motor. Finally, a rotation of this frame around its z axis by θ_2 will result in the desired frame $\{2\}$. The resulting homogeneous transformation can be described as follows

$$^1T_2 = TRAN(z, l_1)ROT(x, 90^o)ROT(z, \theta_2) \tag{2.39}$$

which in matrix form is given by

$$^1T_2 = \begin{pmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.40}$$

The sequence of transformations which moves frame $\{2\}$ into frame $\{3\}$ starts with translation along z axis of frame $\{2\}$ by $-d_2$ units. The distance d_2 represents the mechanical mounting of link 2. This intermediate frame is translated by l_2 units along its x axis so that its origin coincides with the origins of frame $\{3\}$. Rotation by -90^o around x axis of this frame will align its z axis with the z axis of frame $\{3\}$. The joint angle θ_3, by which a rotation around z axis should be performed to arrive to the desired frame $\{3\}$, is determined by the nature of the payload. The $^2T_3$ homogeneous transformation is obtained as

$$^2T_3 = TRAN(z, -d_2)TRAN(x, l_2)ROT(x, -90^o)ROT(z, \theta_3) \tag{2.41}$$

which in matrix form is as follows

$$^2T_3 = \begin{pmatrix} c_3 & -s_3 & 0 & l_2 \\ 0 & 0 & 1 & 0 \\ -s_3 & -c_3 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.42}$$

2.5 Torque Calculations

As stated earlier the torques required can be determined by using the inverse dynamics of the manipulator. Since the iterative N-E (Newton-Euler) algorithm for the inverse dynamics of rigid manipulators [65] requires knowledge of mass distribution of each link with respect to its mass center, it is necessary to provide mass center, mass, and inertia tensor for each link. The inverse dynamics calculations are based on the Newton and Euler equations of motions for each link [27]

$$F_i = m_i \dot{v}_{C_i} \qquad \text{and} \qquad N_i = {}^{C_i}I\dot{\omega}_i + \omega_i \times {}^{C_i}I\omega_i \tag{2.43}$$

These equations imply knowledge of the i-th link inertia tensor ${}^{C_i}I$ calculated for the frame located at the link mass center with identical orientation to the link frame. If the known inertia tensor is for a frame which has a different orientation than the mass center frame, then the *tensor transformation law* and *parallel axis theorem* must be used to calculate ${}^{C_i}I$ for this case.

For the configuration considered in Figure 2.7, a MATLAB program *config.m* has been written to calculate the needed inertia tensors, link masses, mass centers, and distances to origins of the next link. The user has to specify certain options, e.g. link cross-section, mass, length, motor mass, length, etc. Although these input parameters are only estimated at this stage of design, they provide valuable information needed for the N-E iterative algorithm for the inverse dynamic problem.

This algorithm can either be implemented numerically or calculated symbolically. The former approach has efficiency and generality but lacks insight whereas the latter, although error prone, provides excellent insight into the manipulator dynamics. Both methods yielded the same results, thus confirming their correctness. The symbolic approach was implemented in the symbolic manipulation language REDUCE2.

The results shown in Figure 2.8 provide information about the approximate torques for both velocity profiles developed in the preceding section 2.3. The torques are approximate because some of the parameter values were estimated. Friction was not included.

The two differential equations for the 2 DOF manipulator are given below. These are presented as obtained from REDUCE2 without any simplification. It is quite instructional to see the complexity of the dynamics of the rigid manipulator.

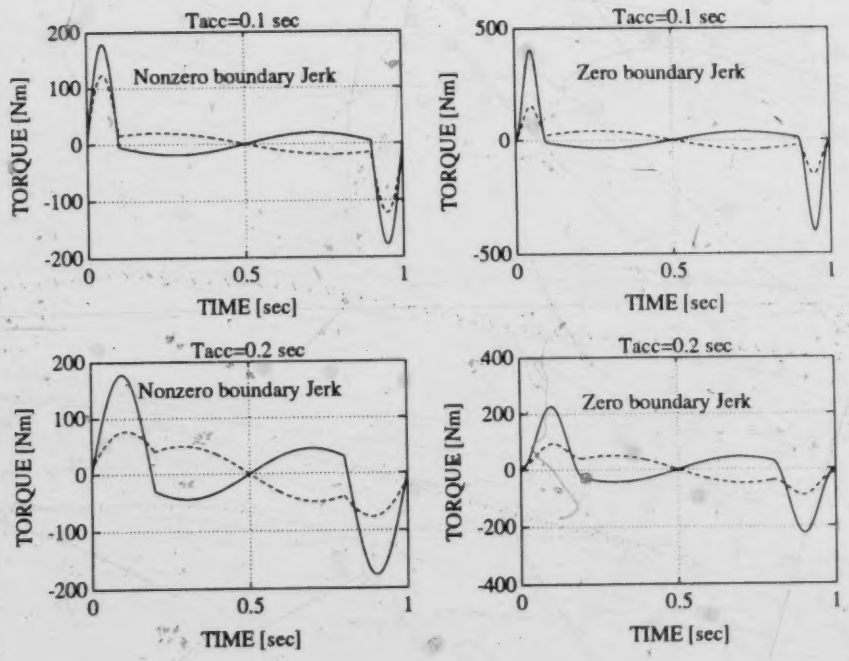


Figure 2.8: Torques for the 2 DOF robotics manipulator.
Solid line - base motor
Dashed line - shoulder motor

The base motor is governed by the following differential equation

$$
\begin{aligned}
\tau_1 \;=\;& gm_3 s_3^2 s_2 c_2 p_{23z} + gm_3 c_3^2 s_2 c_2 p_{23z} \\
& - gm_3 s_2 c_2 p_{23z} + m_3 \theta_2'' p_{C3x}^2 s_3 c_3 s_2 + m_3 \theta_2'' p_{C3x} s_3 s_2 p_{23x} \\
& - m_3 \theta_2'' p_{C3x} c_3 s_2 p_{23z} - m_3 \theta_2'' s_2 p_{23z} p_{23x} + m_3 (\theta_2')^2 p_{C3x}^2 s_3 c_3 c_2 \\
& - m_3 (\theta_2')^2 p_{C3x} s_3^2 c_3 c_2 p_{23z} + m_3 (\theta_2')^2 p_{C3x} s_3 c_2 p_{23x} - m_3 (\theta_2')^2 p_{C3x} c_3^2 c_2 p_{23z} \\
& - m_3 (\theta_2')^2 s_3^2 c_2 p_{23z} p_{23x} - m_3 (\theta_2')^2 c_3^2 c_2 p_{23z} p_{23x} + m_3 \theta_2' p_{C3x}^2 s_3^2 s_2 c_2 \theta_1' \\
& - m_3 \theta_2' p_{C3x}^2 c_3^2 s_2 c_2 \theta_1' - m_3 \theta_2' p_{C3x}^2 s_2 c_2 \theta_1' - m_3 \theta_2' p_{C3x} s_3^2 s_2 c_2 \theta_1' p_{23z} \\
& - m_3 \theta_2' p_{C3x} s_3^2 c_3 s_2 c_2 \theta_1' p_{23x} - m_3 \theta_2' p_{C3x} s_3 c_3^2 s_2 c_2 \theta_1' \\
& - m_3 \theta_2' p_{C3x} c_3^2 s_2 c_2 \theta_1' p_{23x} - 3 m_3 \theta_2' p_{C3x} c_3 s_2 c_2 \theta_1' p_{23x} - 2 m_3 \theta_2' s_3^2 s_2 c_2 \theta_1' p_{23x}^2 \\
& - 2 m_3 \theta_2' c_3^2 s_2 c_2 \theta_1' p_{23x}^2 - m_3 p_{C3x}^2 s_3^2 c_2^2 \theta_1'' + m_3 p_{C3x}^2 s_3^2 \theta_1'' \\
& + m_3 p_{C3x}^2 c_2^2 \theta_1'' + m_3 p_{C3x} s_3^2 c_2^2 (\theta_1')^2 p_{23x} - m_3 p_{C3x} s_3^2 c_2 (\theta_1')^2 p_{23x} \\
& + m_3 p_{C3x} s_3 c_3^2 c_2^2 (\theta_1')^2 p_{23x} - m_3 p_{C3x} s_3 c_3^2 c_2 (\theta_1')^2 p_{23x} - m_3 p_{C3x} s_3 c_2^2 (\theta_1')^2 p_{23x} \\
& + m_3 p_{C3x} s_3 c_2 (\theta_1')^2 p_{23x} - 2 m_3 p_{C3x} s_3 \theta_1'' p_{23z} + 2 m_3 p_{C3x} c_3 c_2^2 \theta_1'' p_{23x} \\
& - m_3 s_3^2 c_2^2 (\theta_1')^2 p_{23z} p_{23x} + m_3 s_3^2 c_2^2 \theta_1'' p_{23z}^2 + m_3 s_3^2 c_2^2 \theta_1'' p_{23x}^2 \\
& + m_3 s_3^2 c_2 (\theta_1')^2 p_{23z} p_{23x} - m_3 c_3^2 c_2^2 (\theta_1')^2 p_{23z} p_{23x} + m_3 c_3^2 c_2^2 \theta_1'' p_{23z}^2 \\
& + m_3 c_3^2 c_2^2 \theta_1'' p_{23x}^2 + m_3 c_3^2 c_2 (\theta_1')^2 p_{23z} p_{23x} + m_3 c_2^2 (\theta_1')^2 p_{23z} p_{23x} \\
& - m_3 c_2^2 \theta_1'' p_{23z}^2 - m_3 c_2 (\theta_1')^2 p_{23z} p_{23x} + m_3 \theta_1'' p_{23z}^2 \\
& + \theta_2'' I_{3yy} s_3 c_3 s_2 - \theta_2'' I_{3xx} s_3 c_3 s_2 - \theta_2'' p_{C2z} p_{C2x} s_2 m_2 \\
& - \theta_2'' s_2 I_{2zz} - I_{3zz} \theta_2' s_3^2 s_2 c_2 \theta_1' - I_{3zz} \theta_2' c_3^2 s_2 c_2 \theta_1' \\
& - I_{3zz} \theta_2' s_2 c_2 \theta_1' + I_{3zz} c_2^2 \theta_1'' + (\theta_2')^2 I_{3yy} s_3 c_3 c_2 \\
& - (\theta_2')^2 I_{3xx} s_3 c_3 c_2 - (\theta_2')^2 p_{C2z} p_{C2x} c_2 m_2 - (\theta_2')^2 c_2 I_{2xx} \\
& + 2 \theta_2' I_{3yy} s_3^2 s_2 c_2 \theta_1' + 2 \theta_2' I_{3xx} c_3^2 s_2 c_2 \theta_1' - 2 \theta_2' I_{2yy} s_2 c_2 \theta_1' \\
& - 2 \theta_2' p_{C2x}^2 s_2 c_2 \theta_1' m_2 + 2 \theta_2' s_2 c_2 \theta_1' I_{2xx} + I_{1zz} \theta_1'' \\
& - I_{3yy} s_3^2 c_2^2 \theta_1'' + I_{3yy} s_3^2 \theta_1'' - I_{3xx} c_3^2 c_2^2 \theta_1'' \\
& + I_{3xx} c_3^2 \theta_1'' + I_{2yy} c_2^2 \theta_1'' + p_{C2z}^2 \theta_1'' m_2 \\
& + p_{C2x}^2 c_2^2 \theta_1'' m_2 - c_2^2 \theta_1'' I_{2xx} + \theta_1'' I_{2xx}
\end{aligned}
\tag{2.44}
$$

The differential equation for the shoulder motor is

$$
\begin{aligned}
\tau_2 \;=\;& gm_3 p_{C3x} c_3 c_2 + gm_3 c_2 p_{23x} + g p_{C2x} c_2 m_2 \\
& + m_3 \theta_2'' p_{C3x}^2 c_3^2 + 2 m_3 \theta_2'' p_{C3x} c_3 p_{23x}
\end{aligned}
\tag{2.45}
$$

$$
\begin{aligned}
&+m_3\theta_2''p_{23x}^2 + m_3p_{C3x}^2 s_3c_3s_2\theta_1'' \\
&+m_3p_{C3x}^2c_3^2s_2c_2(\theta_1')^2 + m_3p_{C3x}s_3s_2\theta_1''p_{23x} \\
&+2m_3p_{C3x}c_3s_2c_2(\theta_1')^2p_{23x} - m_3p_{C3x}c_3s_2\theta_1''p_{23z} \\
&+m_3s_2c_2(\theta_1')^2p_{23x}^2 - m_3s_2\theta_1''p_{23z}p_{23x} \\
&+\theta_2''I_{3yy}c_3^2 + \theta_2''I_{3xx}s_3^2 \\
&+\theta_2''p_{C2x}^2m_2 + \theta_2''I_{2zz} \\
&+I_{3zz}s_3^2s_2c_2(\theta_1')^2 + I_{3zz}c_3^2s_2c_2(\theta_1')^2 \\
&-I_{3yy}s_3^2s_2c_2(\theta_1')^2 + I_{3yy}s_3c_3s_2\theta_1'' \\
&-I_{3xx}s_3c_3s_2\theta_1'' - I_{3zz}c_3^2s_2c_2(\theta_1')^2 \\
&+I_{2yy}s_2c_2(\theta_1')^2 - p_{C2z}p_{C2x}s_2\theta_1''m_2 \\
&+p_{C2x}^2s_2c_2(\theta_1')^2m_2 - s_2c_2(\theta_1')^2I_{2xx} \\
&-s_2\theta_1''I_{2xz}
\end{aligned}
$$

Both equations are non-linear. When only slow motions are considered certain terms can be neglected with an understanding that the model must be inaccurate. However, these equations are less complex than the equations describing behavior of a manipulator with flexible links or joints. Such manipulators are described by a system of partial differential equations which can be solved only numerically using many simplifying assumptions.

From Figure 2.8 one can conclude that actuators of the manipulator should be capable of delivering torques in the range 100 - 200 Nm. This will allow for the uncertainty in the estimated parameter values and also the need to take link flexibility into account.

The choice of actuators (motors) is discussed in the next section.

2.6 Choice of Actuators

Several factors influence the choice of actuators for a robot manipulator. The speed of motion of robotic manipulators is low but the torque requirements may be high. Low-torque high-speed motors are used in conjunction with gears in many robotic manipulators. The typical gear ratios vary between from 80:1 to 200:1. The use of gears has some disadvantages. The transmission efficiency is in the range of 60% to 80% and the friction effects in gears are nonlinear. A more serious drawback is the backlash due to gear teeth. Backlash becomes critical in the final stages of

the manipulator motion. During the final position control phase, when the actuator performs many small motions in both directions, two effects are possible. If the corrective motion is smaller than the backlash then no change of output results and the control algorithm must perform additional corrections which may eventually lead to instability or limit cycle. If the corrective motion exceeds the gear backlash and changes direction then the initial contact of teeth results in a pulse train which excites vibrational modes.

There are, of course, gear mechanisms like harmonic drives, precision gears, preloaded belts and cables, etc. which minimize [82] the backlash effect, however some additional trade-offs occur, for example higher price, increased mechanical sensitivity, frequent adjustments etc.

Additionally, masking effects of high gear ratios [95] hide manipulator dynamics as perceived at the motor shaft.

An alternative solution is to use direct-drive (DD) motors, i.e. motors which are directly connected to the links. For this project it was decided to use DD motors. The distal link with respect to base is directly connected to the DD motor shaft while the proximal link is connected to its stator. DD motors do not exhibit any backlash. Also, since there is no masking effect, the dynamics of the attached mechanical load is directly projected on the motor shaft.

The final choice is driven by the following considerations:

- Torque-speed characteristic
- Price
- Motor amplifiers and their options for computer interfacing
- Integrated sensors and their accuracy
- Available control modes

There are currently available 3 types of DD motors [8]: DC torque motors, brushless DC torque motors, and variable reluctance motors. The MOTORNETICS *megatorque* DD motors were chosen for our 2 DOF manipulator. These are variable reluctance motors, which partially resemble stepping motors. There are three disadvantages in their design: large torque may exhibit torque ripple, excessive weight, and the non-linear torque-speed characteristic. Their torque/weight ratio is smaller than DD direct current torque motors and brushless D.C. torque motors. Their significant

advantage is that they can sustain large continuous torque for an extended period of time without overheating.

There are some inherent features of the DD *megatorque* motors that must be considered. First of all, it is the large weight of both motors. The base motor weighs 74 kg and the shoulder motor 40 kg. The torque requirements for each motor grow exponentially with each additional DD motor located at distal joint [9] . The main reason is that the previously mentioned motors must deliver torque that should be able to move not only the manipulator links but also the distal joint actuators. This does not create a problem for the base motor because its weight is supported by the mechanical structure on which the manipulator is attached. The base motor of the manipulator described in this thesis is supported by a metal frame fixed to the floor and its shaft is in a vertical position. To minimize the effects of the mass of the shoulder motor on the base motor, the shoulder motor is placed on the vertically pointed shaft of the first DD motor so that its weight is opposed by forces in bearings of the first motor (see Figure 2.9 and Figure 2.10). This has the beneficial effect of increased damping of the first motor since its friction torque has been increased. The disadvantage of this mechanical configuration is that the enlarged friction becomes more significant and must be accounted for in any applied control algorithm.

Each motor has 18 stator poles with 6 coil segments in series. The stator is dual and the rotor moves between both stators. This means the magnetic flux path is short and magnetic losses are small.

DD motors have very small damping ratios due to small viscous friction and it is recommended to control them with velocity feedback to increase the damping ratio. Large required torques at slow velocities mean that large coils must be used resulting in large inductance. This leads to a large electric constant in DD motors.

The closest description of our *megatorque* DD motors is presented by Welburn [105]. *Megatorque* DD motors are 3 phase variable reluctance motors without any permanent magnets. The small inertia rotor is moving between the dual stator. The higher number of motor poles increases torque and reduces power so that $P = \tau\omega$ is constant.

Stator poles and rotor surface are made up from teeth-like structures having the same effect as an additional increase in the number of poles. Since no permanent magnet is used there are no problems with demagnetization and the motor can operate under maximum winding current.

AC power is first rectified and the resulting 150 volts DC is pulse width modulated (PWM) by the motor amplifier in which MOSFET switches operate with 20kHz

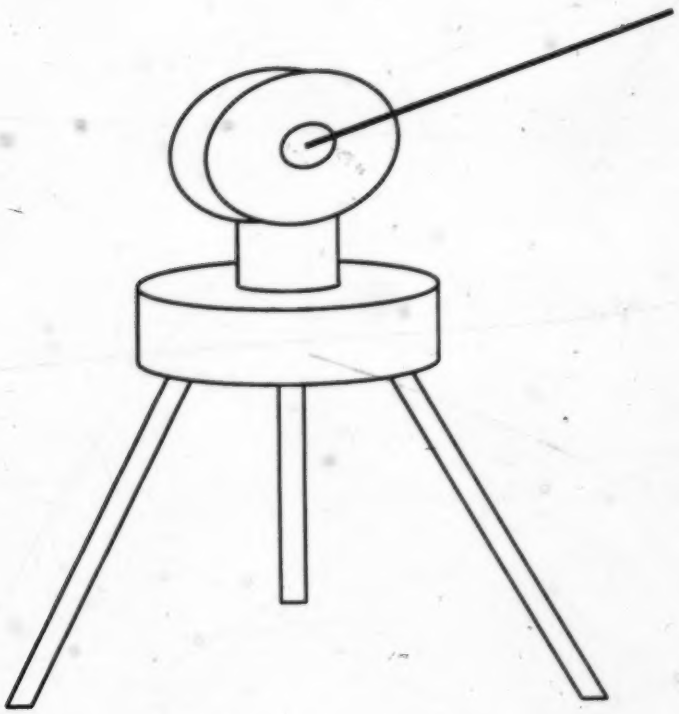


Figure 2.9: Assembly of the 2 DOF manipulator.

frequency in half-bridge configuration.

The double stator has 36 windings which are divided into 2 groups for each stator. At any time instant only 3 out of 18 windings are excited on each stator. To slow down the velocity and to increase the torque, both stators and rotor have 150 teeth which function as motor poles and torque is produced by exciting these tooth poles. There are 150 cycles per each revolution, i.e. one cycle per each tooth. This tooth structure enlarges the motor torque as if a gear ratio 1:150 were used.

A good torque performance is not sufficient to qualify a given motor actuator for robotics application if adequate position and velocity feedback sensors are not available. It is the fact that the position and velocity sensors are integral parts of the *megatorque* DD motors that makes them attractive for some robotics applications. The position sensor is a resolver whose windings are built in the motor and it distinguishes 2^{10} positions between adjacent teeth. The position is available in the incremental quadrature output signal form and provides resolution 614 400 counts/rev. The RDC (Resolve to Digital Converter) distinguishes absolute position only between adjacent teeth. This information is lost when the position is converted into the incremental quadrature signal. The RDC internal information is used to calculate the

Figure 2.10: 2 DOF manipulator and acceleration signal conditioning unit.

shaft velocity which is converted to an analog signal via 8 bit DAC.

Position and velocity available from RDC are used in the motor amplifier 32 bit microprocessor which takes care of proper commutation, smoothing the torque ripple, and linearization of the torque-speed characteristic with frequency 2 kHz.

Maximum power can be achieved at about 0.3 rev/sec and maximum stall torque is 225 [Nm] for the base motor RS1410 and 125 [Nm] for the shoulder motor RS1010, which satisfy the torque requirements of the 2 DOF manipulator.

2.7 Link Choice

The 2 DOF robotics manipulator has two joints, each with a single degree of freedom, and two links. The first link, in our case, is the bracket joining the shoulder motor stator to the shaft of the base motor. This attachment is a heavy rigid structure whose inertia tensor is not known due to its complicated physical form. This first link together with the shoulder motor, the bracket, and the rotor of the base motor provide the main inertia for the base motor. The large inertia has the beneficial effect of increasing the damping ratio of the base DD motor as mentioned in the previous section.

It is the choice of the second link and its flexibility that is crucial for the research described in this thesis. The selection of the material for the flexible link and determination of its geometrical dimensions are the results of compromise among several contradictory requirements.

The design of the flexible link is influenced by the need to have low modal frequencies and to avoid plastic deformations. For this project it is necessary to ensure that the flexible link undergoes only elastic deformations when it is subjected to the maximum possible torque that can occur during a typical motion. It is desirable that the link fundamental frequency is such that a low sampling frequency can be used. This will enable more calculations to be carried out on the control computer between two successive sampling instants. On the other hand if plastic deformations are to be avoided, one should choose a rather thick link with high fundamental modal frequency. It is not unusual for the system to become unstable. Thus the flexible link is exposed to fast changing torque from the maximum value in one direction to the maximum value in the opposite direction. Under such conditions, only a very thick link will not be plastically deformed.

To determine a suitable length and a cross-section of the flexible link, it is necessary to evaluate their quantitative effects on the link modal frequencies. A sym-

metric constant cross-sectional profile was assumed so that both bending vibrations, i.e. in ${}^2Y$ and ${}^2Z$ directions, can be investigated independently. Results valid for the vibrations in ${}^2Y$ direction are applicable for the bending vibrations in ${}^2Z$ direction. Let us denote the displacement due to vibrations in the ${}^2Y$ direction by $v(x,t)$. The link modal frequencies are determined from free vibration, i.e. when no external forces or torques are acting on the link. For a thin link, the Euler-Bernoulli [68] equation describing the free vibrations in the ${}^2Y$ direction has the following form

$$EI\frac{\partial^4 v(x,t)}{\partial x^4} + \rho A\frac{\partial^2 v(x,t)}{\partial t^2} = 0 \tag{2.46}$$

where the product EI is the flexural rigidity of the link, I is the moment of inertia of the cross-sectional area about the centroidal axis of the link, ρ is the link mass density, and A is the link cross-sectional area.

To obtain the solution to free vibrations it is assumed that the displacement $v(x,t)$ is time-space separable and can be expressed as

$$v(x,t) = Y(x)F(t) \tag{2.47}$$

and the periodic nature of the free vibrations implies that the time function $F(t) = Ke^{j\omega t}$ where K is an arbitrary constant. The equation 2.46 can be written as

$$\left\{EI\frac{d^4Y(x)}{dx^4} - \omega^2\rho AY(x)\right\}F(t) = 0 \tag{2.48}$$

Because the time function $F(t)$ is not zero, then the expression in the curly brackets must be identically zero leading to the following differential equation

$$\frac{d^4Y(x)}{dx^4} - \beta^4 Y(x) = 0 \tag{2.49}$$

where

$$\beta^4 = \omega^2\frac{\rho A}{EI} \qquad \Longrightarrow \qquad \omega = \beta^2\sqrt{\frac{EI}{\rho A}} \tag{2.50}$$

The general solution of the equation 2.49 is

$$Y(x) = C_1\sin\beta x + C_2\cos\beta x + C_3\sinh\beta x + C_4\cosh\beta x \tag{2.51}$$

where C_1, C_2, C_3, C_4, and β are constants to be determined. Two special cases [69] which are usually considered to determine the modal frequencies for free vibrations

are described below.

The *pinned-free* case boundary conditions for the pinned end, i.e. at $x = 0$, are in the form of zero deflection and zero bending moment

$$Y(x)_{|x=0} = 0 \quad and \quad \frac{d^2Y(x)}{dx^2}_{|x=0} = 0 \tag{2.52}$$

and the boundary conditions for the free end, i.e. $x = L$, are in the form of zero bending moment and zero shearing force

$$\frac{d^2Y(x)}{dx^2}_{|x=L} = 0 \quad and \quad \frac{d^3Y(x)}{dx^3}_{|x=L} = 0 \tag{2.53}$$

When the above boundary conditions are used in the equation 2.51, the expression for β is

$$\sinh \beta L \cos \beta L = \cosh \beta L \sin \beta L \tag{2.54}$$

The *clamped-free* vibrations have identical boundary conditions for the free end. Since the clamped end cannot move, its boundary conditions are

$$Y(x)_{|x=0} = 0 \quad \text{and} \quad \frac{dY(x)}{dx}_{|x=0} = 0 \tag{2.55}$$

The equation 2.51 and the boundary conditions for the clamped-free vibrations yield the following expression from which β can be calculated numerically

$$\cos \beta L \cosh \beta L = -1 \tag{2.56}$$

Both equation 2.54 and 2.56 can only be solved numerically and they provide infinite number of β's, each corresponding to a different modal frequency.

However, the situation when the link is connected to a motor is neither the pinned-free case nor the clamped-free case. One boundary condition is changed ([42] and [56]), because the link bending moment at the LINK pinned end must be balanced by inertial moments. If I_h represents all inertias, except for the flexible link, acting on the joint then the following boundary condition replaces $\frac{d^2Y(x)}{dx^2}_{|x=0} = 0$ in equation 2.52

$$EI\frac{d^2Y(x)}{dx^2}_{|x=0} + \omega^2 I_h \frac{dY(x)}{dx}_{|x=0} \tag{2.57}$$

	Clamped-Free				Pinned-Free			
L [m]	0.5	1.0	1.5	2.0	0.5	1.0	1.5	2.0
f_1 [Hz]	20.77	5.19	2.31	1.30	91.05	22.76	10.11	5.69
f_2 [Hz]	130.17	32.54	14.46	8.13	295.21	73.80	32.80	18.45
f_1 [Hz]	364.53	91.13	40.50	22.78	615.72	153.93	68.41	38.48

Table 2.3: Modal frequencies for clamped-free and pinned-free vibrations.

This results in the following equation for β.

$$(\cos\beta L \cosh\beta L + 1) + \frac{\rho A}{I_h \beta^3}(\cosh\beta L \sin\beta L - \sinh\beta L \cos\beta L) = 0 \qquad (2.58)$$

The above equation demonstrates that if the I_h inertia is small, then the modal frequencies will approach the frequencies of the pinned-free case. On the other hand, if I_h is large then the the frequencies will be close to those obtained in the clamped-free case. In our case, the inertia I_h is not known and only the clamped-free and pinned-free modal frequencies can be calculated.

Taking into consideration the above contradictory constraints, the following link parameters were chosen: material high-yield steel, 1/4" square cross-section ($\approx 6.35mm$). The link mass is approximately 0.5 kg. The full square cross-section results in lower modal frequencies than a link of the same mass and length with a hollow cross-section. Additional advantage of the full square cross-section is that it exhibits much smaller torsional deformations than an equivalent link with circular cross-section. The 1/4 inch dimension is the minimum required width for convenient mounting of the commercially available strain gages.

To determine the range of the first modal frequencies given the above choice, only the link length is left as the variable which must be varied to obtain the desirable low modal frequencies. The range of the modal frequencies is determined by the two limit cases, i.e. clamped-free and pinned-free cases. The clamped-free case results in the lowest modal frequencies for a given link, while the pinned-free case exhibits the highest modal frequencies. Calculations for several different lengths were tried while respecting the space available in the robotics laboratory. The results are summarized in the Table 2.3. Based on the results shown in Table 2.3, the length of 1.5 m was chosen for the flexible.

Chapter 3

Manipulator Hardware Design

3.1 Introduction

Several phases are involved in the design of the robotics test-bed. The previous chapter deals with the mechanical design of the manipulator. This chapter describes the development of electronic hardware and associated software needed to control the flexible manipulator.

A schematic diagram of the 2 DOF manipulator hardware is shown in Figure 3.1. Besides the hardware, which is an integral part of the motors, the system consists of the following :

- A host computer (80386 PC) with four T800 transputers.
- Digital and analog I/O interface boards.
- Sensors (strain gages and accelerometer) and associated electronic hardware.
- Mechanical brakes and limit switches for the motors.
- 3 phase power supply system for the motors.

Section 3.2 describes the Quadrature Decoder PC board developed to measure shaft positions of both motors. The design of the digital I/O interface box is described in section 3.3. Strain gages and the associated amplifiers are discussed in section 3.4. The design of the motor power distribution and the interruptable power supply for emergency brakes is described in section 3.5. The design of 4-th order Bessel antialiasing filters with modifiable corner frequencies is explained in section 3.6. Section 3.7 deals with the design of a controlled double integrator to obtain velocity and position of the tip of the flexible link from acceleration measurements.

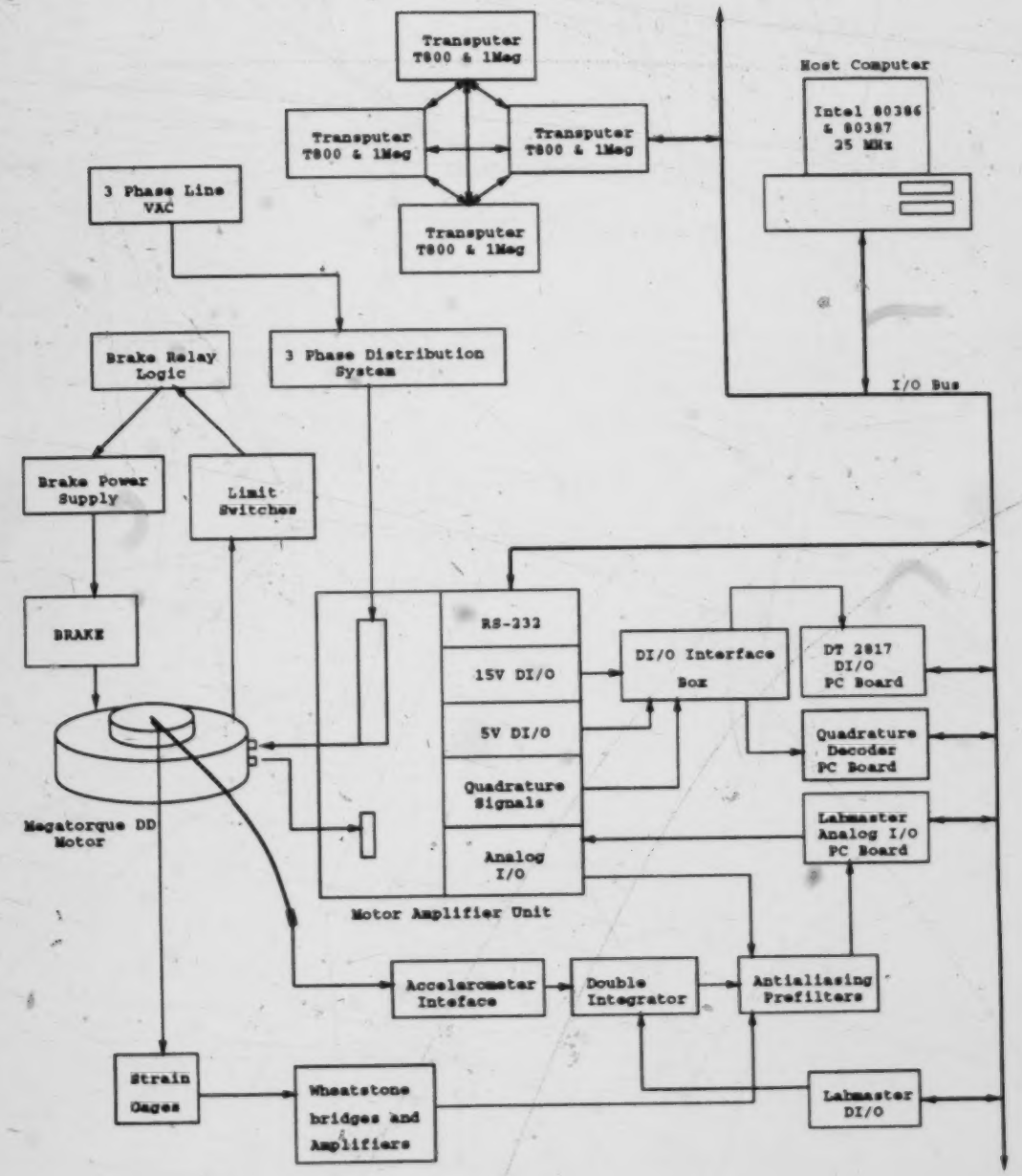


Figure 3.1: 2 DOF robotics manipulator hardware diagram.

To control or efficiently use some of the hardware subsystems, complicated and often cumbersome software algorithms must be employed. This is inconvenient for a researcher using the robotics test-bed. To eliminate this inconvenience, software drivers were developed. Section 3.8.1 describes the software driver controlling the generation of a timer hardware interrupt that specifies the sampling frequency of tested control algorithms. The software driver for the analog to digital conversion (ADC) is explained in section 3.8.2. Reading of the correct motor shaft positions is controlled by the software driver described in section 3.8.3.

The conceptual development of the manipulator test-bed, the design of the various subsystems described above and in chapter 2 and the construction of the complete system required two years of effort. The task was especially difficult because of the pioneering nature of the project. To the best knowledge of the author reports of similar efforts are not available in the published literature.

3.2 Quadrature Decoder Board

The information about the motor shaft position is available in two forms. The first option uses RS-232 serial communication protocol. This method is slow and can only be used for simple tests of the motor functionality. It is not suitable for direct control purposes. However, it has proved useful in testing the PC quadrature decoder board (QDB) discussed later in this section.

The second option for obtaining the motor shaft position is in the form of quadrature signals from the motor amplifier. The pulses, which are proportional to the shaft position, come from the Resolver to Digital Convertor (RDC) on two signal conductors. To distinguish the direction of the motion, the $\pm 90^{\circ}$ phase shift between the pulses on both conductors is utilized.

Since the *megatorque* DD motors provide resolution of 614400 pulses/revolution, a 19 bit counter is needed ($2^{19} = 524288$) to cover 5.35 radians working envelope for a single joint. At the time of the manipulator design there was, to the author's knowledge, no PC quadrature decoder board available. Only recently a VME board with 16 bit counters appeared on the market.

Although the counter content could be associated with hardware interrupts so that software could keep track of the position information, there is only a limited number of hardware interrupts available and the design should preserve them for other uses. It is desirable to avoid possible hardware interrupt conflicts which are difficult to predict at early design stages. Additionally, it is convenient if the hardware can

take care of the motor shaft position completely so that correct shaft position readings can be obtained instantaneously using simple I/O instructions.

The following design specifications were defined for the QDB :

- It must be fully compatible with PC-AT I/O bus.
- The address decoding circuit must provide an option for changing the board address.
- It must generate a specified number of wait states so that it can be used with different I/O bus clock frequencies.
- The TTL level output from the QDB to the I/O bus must be driven by tri-state logic output signals.
- It must perform the following minimum set of functions:
  - Clear counters.
  - Read content of both counters simultaneously into a tri-state buffer.
  - Read content of buffers - this is same as reading both motor shaft positions simultaneously.
- Input quadrature signals must go through a Schmitt-trigger circuit to improve timing of rising and falling edges during motor slow motions when the transition between well defined logical levels may cause incorrect position readings.
- 24 bit up/down counters must work in $2's$ complement code. To speed up position reading, content of the counters must be expanded to 32 bits that are needed for long integer number representation.
- The incoming quadrature signals must be filtered so that spike noise is eliminated and the counter contents hold the correct shaft position.
- The quadrature signal cables coming from the motor amplifier must be shielded.

The quadrature signals are available on the 50 pin CN2 connector together with many other digital signals of different voltage levels. All connectors on the motor amplifier are close to 3 phase 220 VAC power which represents a considerable noise source for all low level signals in its proximity. A shielded twisted pair signal cable is used for the CN2 connector to minimize cross-talk and other interference.

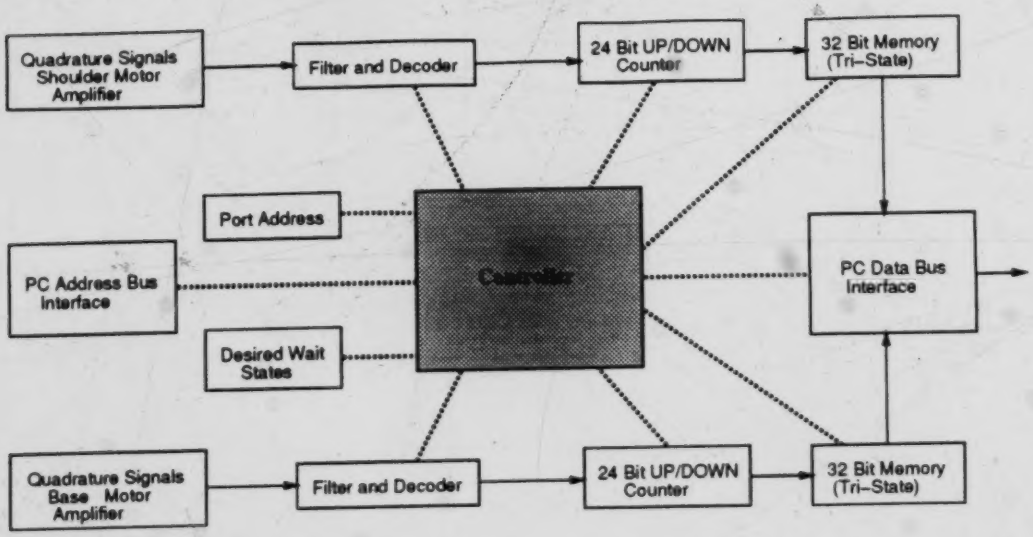


Figure 3.2: Block diagram of the QDB PC board.

To minimize the development time, the design of more complicated subsystems was carried out using Electronically Programmable Logic Devices (EPLD) ([2] and [3]). The main advantage in using EPLD for the design of logical systems is that subsystems allocated to a particular EPLD can be easily modified, their functions can be simulated and tested, and if the actually burnt design does not satisfy its requirements, the EPLD chip can be erased by exposure to ultraviolet light and the design can be modified and repeated.

Figure 3.2 shows the block diagram of the QDB. The QDB board converts the quadrature signals coming from the base and the shoulder motors into 32 bit binary representation. The resulting 32 bit integer numbers, which represent the shaft positions, are available to the host 386 computer. The lines marked with arrow heads in Figure 3.2 describe processing of the position information while the dotted lines represent control signals needed for correct function of the QDB. The detailed description of the QDB can be found in Appendix A.1. The explanation of the main subsystem of the QDB follows.

Filter and Decoder subsystems for each motor are located in one EPLD900 device. The digital filter eliminating spike noise was implemented as outlined in Hewlett Packard HCTL-2000 Quadrature Decoder/Counter Interface integrated circuit (IC). The filtered quadrature signals are then used by the decoder circuit which evaluates valid pulses and generates signals that control the following 24 bit Up/Down counter.

A *24 Bit Up/Down Counter* subsystem is implemented in another EPLD900

device, one for each motor. The main function of the 24 bit subsystem is to count pulses in both directions and keep track of the motor position.

32 Bit Memory is a subsystem which samples the content of the 24 bit Up/Down counter on request from the 386 CPU. Since the 24 bit Up/Down counter follows the motor motion, the simultaneous reading of the contents of both 24 bit Up/Down counters guarantees that the 32 bit memory holds position information from both motors taken at the same time instant.

PC Address Bus Interface and *PC Data Bus Interface* are subsystems that detect all requests from the host 386 CPU and place the desired results on the I/O bus. *Port Address* and *Desired Wait States* subsystems ensure that the QDB avoids I/O bus address conflicts and accommodates the I/O bus speed.

Controller subsystem ensures that the QDB timing and function specifications are met. The QDB complies with the I/O bus specifications.

Testing is an integral part of any development and in the case of the QDB two different sets of tests were performed. The first set consisted of tests with externally generated quadrature signals. Since it was not possible to locate a pulse generator that would generate quadrature signals, another PC machine with DI/O (Digital Input/Output) board was used to generate the desired signals. Although the program controlling the generation of the quadrature signals was written in assembler language, the maximum frequency of the *channel A* and *channel B* signals was about 220 kHz, which corresponds approximately to 1 RPS. One experiment, for example, consisted of generation 10^7 quadrature steps in clockwise direction followed by 10^7 steps in the counter clockwise direction. At the end the counter contained the proper zero result. The second set of the tests was performed with the DD motors and their interface. The RS-232 interface was used to control the DD motors and to read the shaft positions. The positions available via RS-232 were compared with the positions from the QDB. The maximum observed difference was in the range from 1 to 3 pulses. Both tests conclusively showed that the QDB performs well. The QDB board is shown in Figure 3.3 and Figure 3.4.

3.3 Digital I/O Interface

The digital I/O subsystem, which provides the interface between the motor digital input/output (DI/O) and PC DI/O and QDB, is described in this section. The *megatorque* DD motors can be used in several modes (torque control mode, position control mode, velocity control mode, memory stored motion mode, etc.) that require

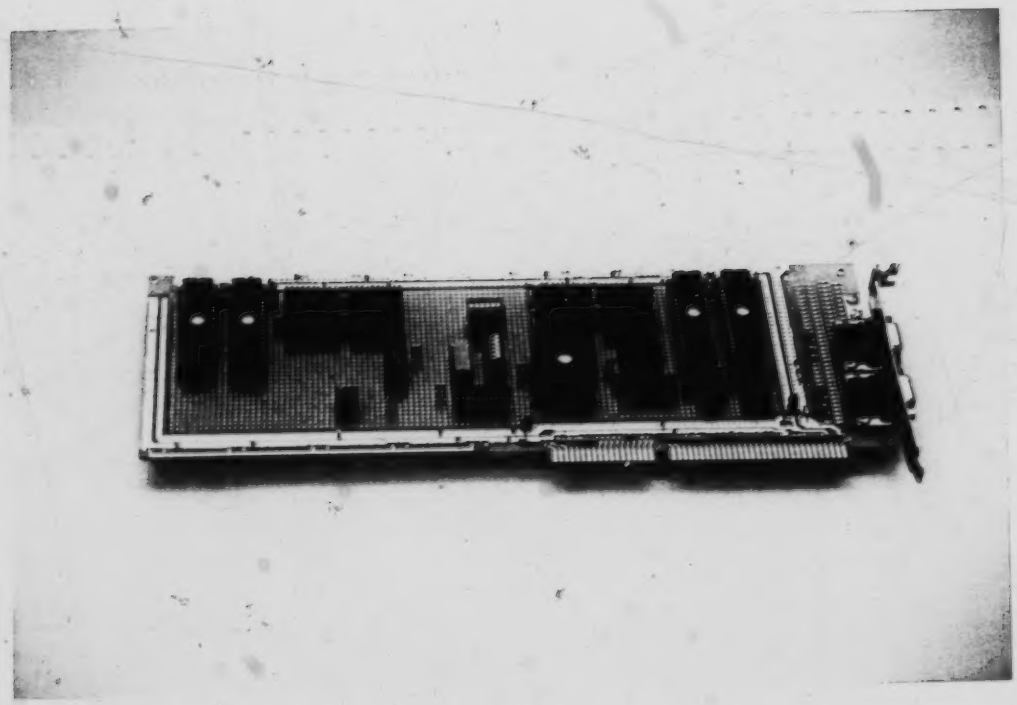


Figure 3.3: Quadrature decoder board (component side).

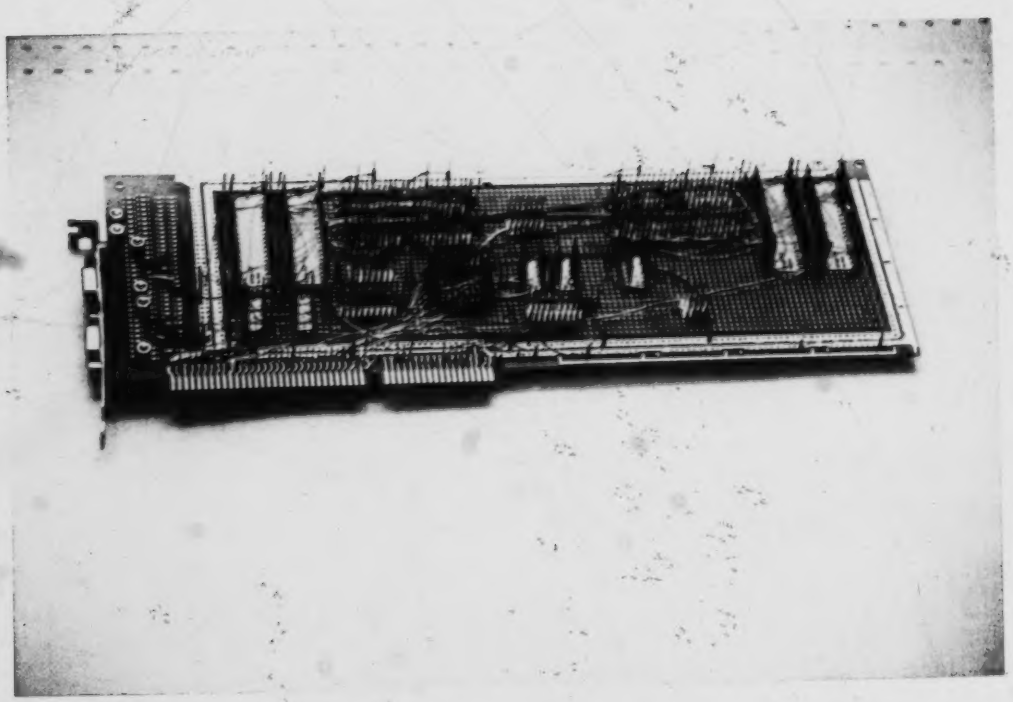


Figure 3.4: Quadrature decoder board (wire-wrapped side).

specific input and output digital signals for their functions. Although some functions associated with these signals can be carried out with RS-232 interface, the digital signals are faster and certain subsets of signals cannot be substituted by the RS-232 commands. It was essential to design digital I/O (DI/O) boxes to interface the computer and the DD motor DI/O subsystems. The computer side is characterized by TTL levels while the motor amplifier side has a variety of voltage levels ($12 \simeq 24V, 5V$, etc.). To minimize the danger of computer or motor amplifier damage if a short circuit occurred, separate power supplies were provided for the motor and computer sides and opto-couplers were used for electrical isolation of motor and computer sides. The digital I/O interface box is shown in Figure 3.5. More details can be found in Appendix A.2.

3.4 Strain Gages and Strain Gage Amplifiers

In the 2 DOF robotics manipulator the position and velocity of the two motor shafts can be sensed by the synchro resolver. Note that the resolver does not provide any information about the tip position and the vibrations of the flexible link.

To measure the flexible link vibrations, strain gages are proposed. It is convenient to locate a strain gage close to the motor shaft where the maximum strain of the flexible link exists.

There are several advantages in using strain gages. They provide direct measurements of vibrations without any interference due to the underlying rigid motion. They have a very short time constant, hence any delay due to measurements can be neglected. They are reasonably priced.

The disadvantages of strain gages are measurement drift and a frequent need for recalibration. Their dynamic measurements depend on their location due to the modal nature of vibrations. However, this position dependence is not critical if they are used for compensation of steady-state bending effects due to gravity.

For the strain gage amplifier, a design used in another research project [94] was modified. The strain gage amplifiers are shown in Figure 3.6. The details are given in Appendix A.3.

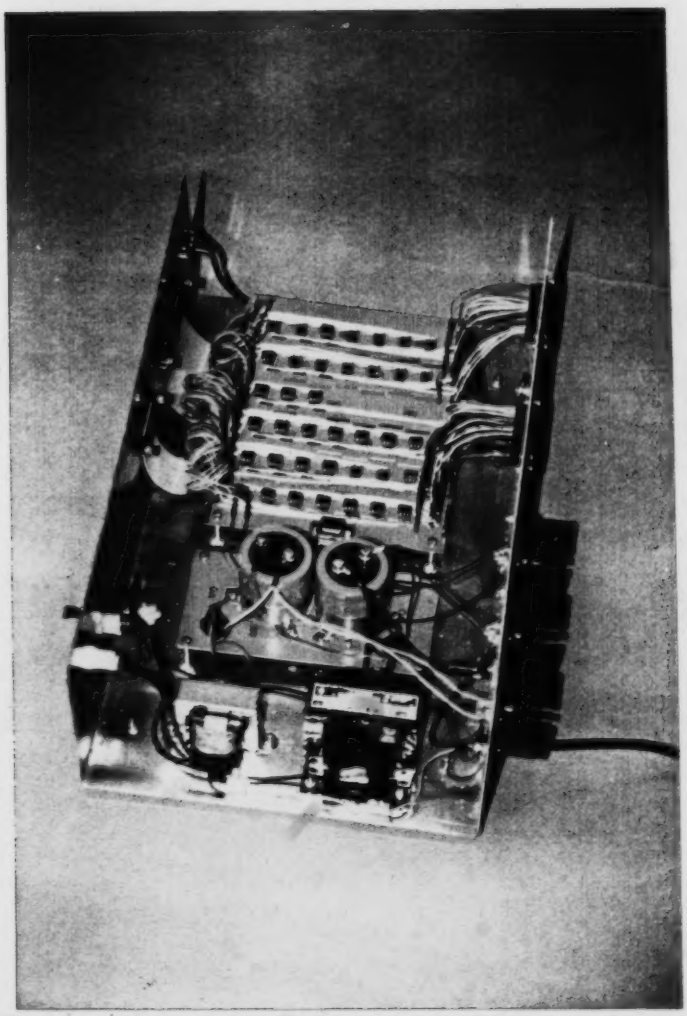


Figure 3.5: Digital I/O box and its power supplies.

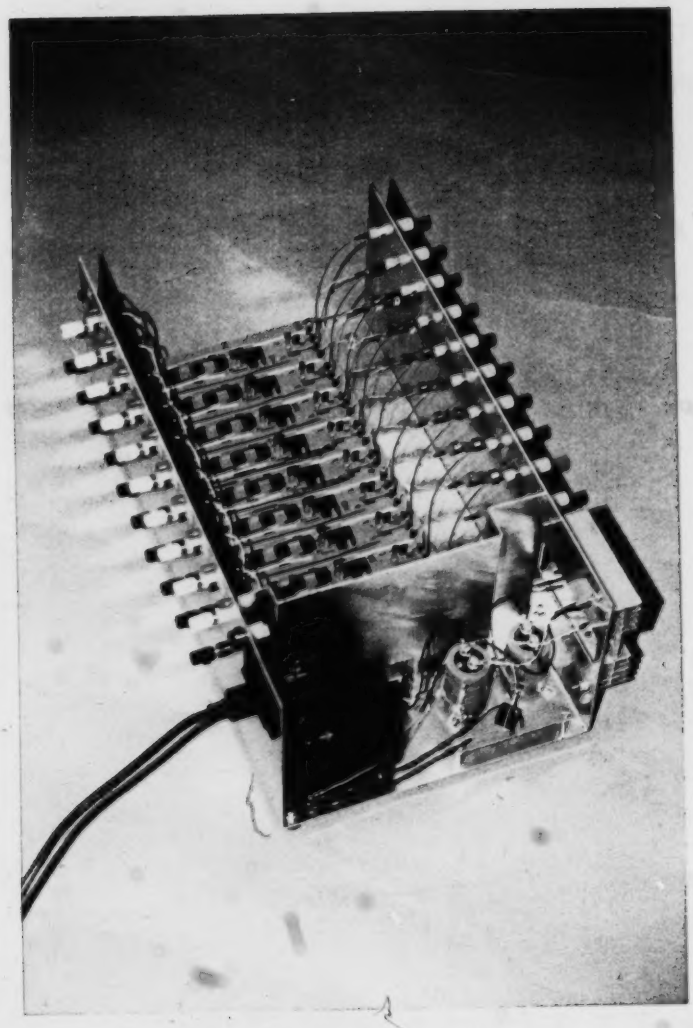


Figure 3.6: Strain gage amplifiers and power supply.

3.5 3 Phase Power Distribution and Motor Brake Activation

Both *megatorque* DD motors require 3 phase 220 VAC power. The following power distribution requirements were specified:

- The ability to start and stop the 3 phase power distribution must be retained without wear-off of the final power switch.
- An alarm push-button must be available for safety and emergency.
- 3 phase power must be also controlled from limit switches defining the robot workspace.

The DD motors are equipped with brakes which require a 24 VDC-3A power supply. Although the power supply could be purchased, it was designed and built because there is an additional need to interrupt the power supply and initiate braking whenever the 2 DOF robotics manipulator reaches the boundary of its workspace.

The safety mechanism, which controls both brakes and the 3 phase power distribution, can be activated either by the user or by the limit switches located at the boundary of the flexible manipulator workspace. The schematic diagram is shown in Figure C.1 in Appendix C.

It is desirable to ensure that disconnecting the 3 phase power distribution will activate the brakes, and reciprocally, the brake activation should disconnect the 3 phase power. The reason is that braking is more efficient if zero torques are developed.

If the 2 DOF robotics manipulator goes outside its workspace, one or both of the limit switches remain open and the corresponding brake is locked and the brake power supply is disconnected. To unlock the brake to return the robotics manipulator back to its workspace, the brake power supply must be equipped with an additional voltage output, which is independent of the limit switches, used only for returning the robotics manipulator back to its workspace. The 3 phase power distribution and the brake control system are shown in Figures 3.7 and 3.8.

3.6 Antialiasing Filters

Antialiasing filters are used to suppress high frequency signal components that, due to sampling, would be folded into the low frequency components and distort the

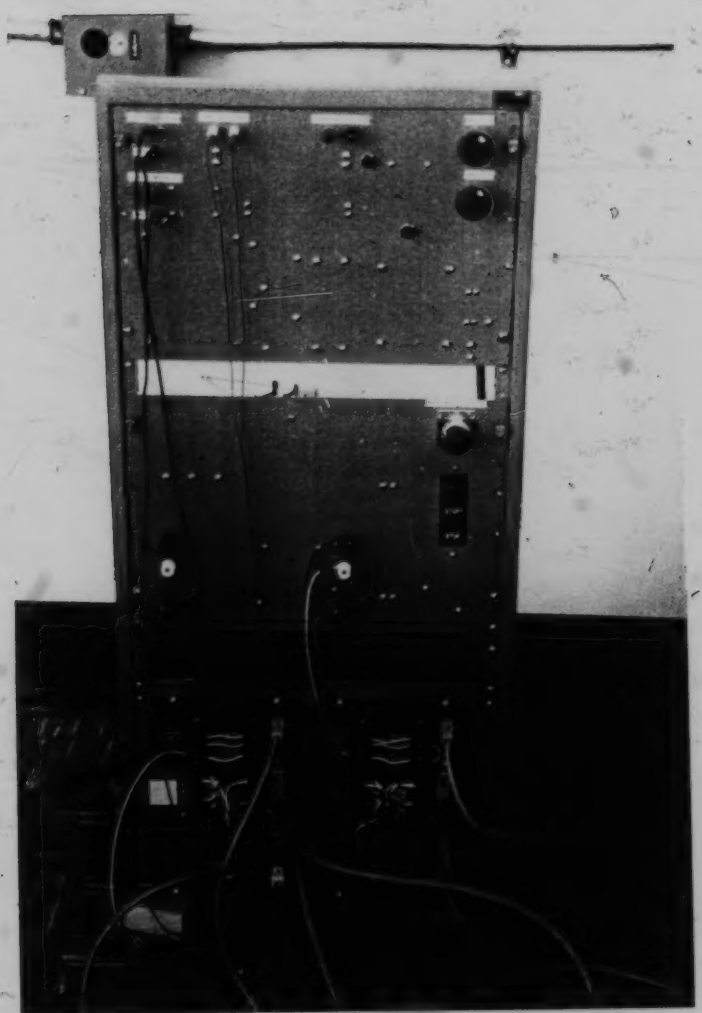


Figure 3.7: Brake control system (top), 3 phase distribution (middle), and direct-drive motor amplifiers (bottom) - front view.

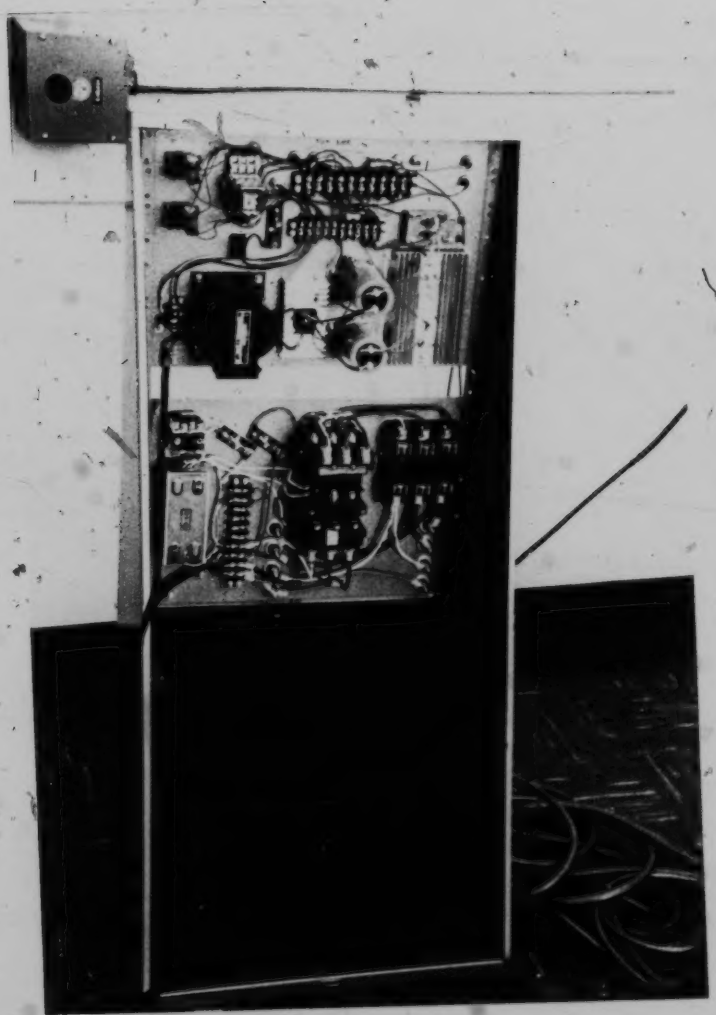


Figure 3.8: Brake control system (top), 3 phase distribution (middle), and direct drive motor amplifiers (bottom) - rear view.

useful low frequency signal [10]. This is particularly important when flexible systems exhibiting an infinite number of modal frequencies are controlled.

In high performance control applications two considerations governing use of antialiasing filters must be considered:

- Efficient attenuation of higher frequencies beyond the folding frequency.
- Distortion of the signal passing through the antialiasing filter.

If attenuation were the main consideration then Chebyshev or elliptic filters would be a good choice because they have high attenuation even for low order low-pass filters. Signal distortion plays a significant role in control applications because it is desirable to know the correct time shape of the sampled signal. This means that low-pass filters with a linear phase are a good choice for antialiasing filters in control applications. Such filters delay equally all frequency components below a certain maximum frequency after which the phase starts to loose its linearity.

Bessel filters are an example of linear phase filters with a trade-off in attenuation performance. The linear phase property is compromised by smaller attenuation beyond the filter corner frequency, which means that higher order Bessel filters must be used to achieve better attenuation. To compromise between hardware complexity and the filtering requirements, a fourth order Bessel filter is chosen. The 4-th order Bessel filters are shown in Figure 3.9. The details of design of the 4-th order Bessel antialiasing filter are given in Appendix A.4.

3.7 Double Integrator

If measurements of the acceleration of the tip of the flexible link are available, then double integration can be made to determine the velocity and position of the tip. Although numerical integration is attractive due to the ease of implementation and modification, it was not pursued because good accuracy requires fast sampling and that implies a significant computer burden. Therefore analog integration is chosen as a viable alternative.

The primary difficulty with analog integration is the saturation nonlinearity associated with all analog circuits. To circumvent this, it is necessary to discharge the feedback capacitor before the output voltage approaches the saturation level. The simplified circuit diagram of the double integrator can be seen in Figure 3.10.

Since during the discharging process no integration should occur, switches SW1 and SW3 are open. It is conceivable to connect SW1 and SW3 to the ground

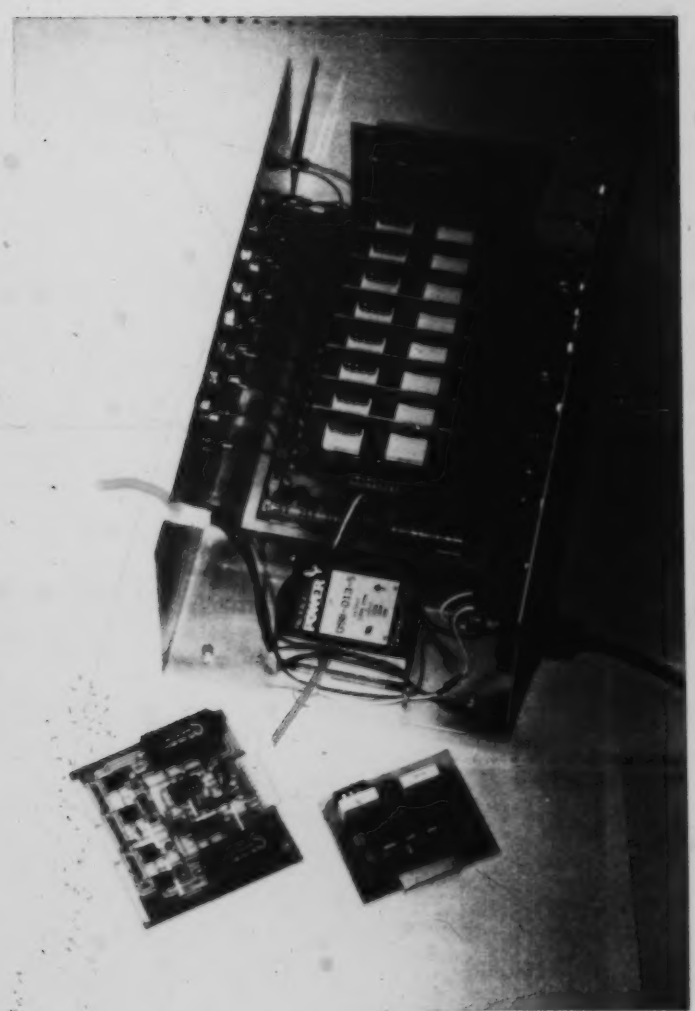


Figure 3.9: Bessel filter and double integrator box. The 4-th order Bessel filter board (bottom right) and the double integrator board (bottom left).

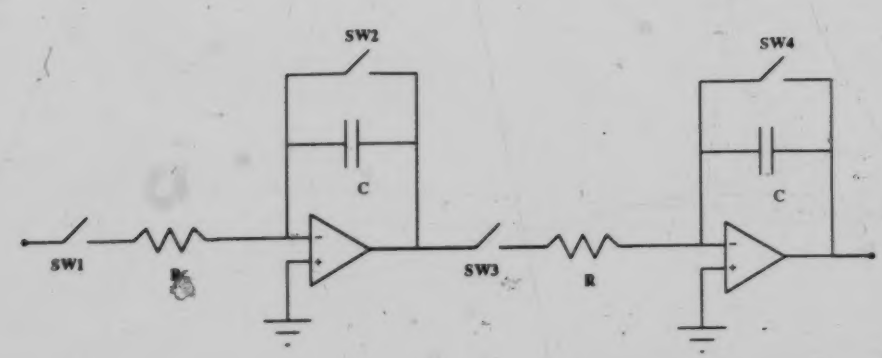


Figure 3.10: Double integrator.

instead of having them open but it is not a good solution because it would create a short circuit for the sensor output.

Figure 3.10 is only a conceptual design. The double integrator is shown in Figure 3.9. The details of the design are given in Appendix A.5.

3.8 Software Drivers

The main reasons for the development of software drivers are to efficiently use hardware devices and to hide the low level program control of hardware devices.

While it is easier to program the hardware devices in C programming language, programming in assembler language is preferred when speed and efficiency are the priorities. It must be emphasized that assembler language programming is difficult. It lacks software portability and exhibits maintenance problems. The optimum approach seems to be to develop software in C and once the C code is stable and does not undergo any further changes, the assembler programming is undertaken. The protocol for software interface between C and assembler must be carefully followed especially when parameters or results are transferred.

Programming of software drivers exhibits difficulty not found in standard task programming. If hardware devices are not correctly controlled they may halt the computer. It is impossible to determine the cause because the only recovery is to reboot the computer. This leaves the programmer without any opportunity to analyze the problem. It requires a lot of deduction and engineering intuition to debug the low level software drivers when no special software and hardware facilities are available. The difficulty is frequently due to inadequate documentation of hardware devices. The remainder of this chapter discusses software drivers for the hardware timer interrupt

and data acquisition.

3.8.1 Hardware Timer Interrupt

Among all subsystems available on the Lab Master PC board, the 9513A hardware counter-timer is the most difficult to program and control. Programming of the 9513A hardware counter-timer is difficult because its initialization is complicated because of complex addressing and sensitivity to the sequence of operations to access specific registers. One of the main goals of this software driver is to make the initialization transparent to the user except for choosing the sampling period T. Details are provided in Appendix A.6

3.8.2 Analog to Digital Conversion

Analog to digital conversions of signals occurs periodically and are driven by the programmed hardware timer interrupts. Since the time available between consecutive interrupts is limited, this software driver was written with the goal to minimize the ADC time.

The user can call this assembler driver directly from his C program and only needs to specify the initial channel number, the number of subsequent channels to be converted, and a pointer to the integer array where the results must be stored. The entire initialization, delay loops, addressing, etc. are transparent to the user. Details are given in Appendix A.7.

3.8.3 Motor Shaft Position

This software driver reads the the motor shaft positions using QDB. It has been optimized for time efficiency. The calling C program provides two pointers to long integers and on return both long integers contain the updated shaft positions. The implementation details are given in Appendix A.8.

Chapter 4

Assessment of Test-bed Performance

4.1 Introduction

The purpose of this chapter is to discuss various tests which were performed on the 2 DOF manipulator test-bed which is shown in Figure 4.1.

The purpose of the tests were as follows :

- to check if the experimental test bench performs satisfactorily
- to determine as accurately as possible the different components of friction of the motor, motor time constants, delays, etc.
- to demonstrate the need for friction compensation in any control scheme
- to investigate frequency response of the motor
- to determine the acceleration ranges so that appropriate accelerometers could be purchased
- to develop methods to interpret properly results from the double integrator
- to measure the flexible link modal frequency using fast Fourier transform (FFT)

This chapter is organized as follows. The components of DD motor friction as well as the measurements of the stiction and the Coulomb friction are described in section 4.2. Section 4.3 presents PID control law with different friction compensation schemes to control the motor shaft position. The combined effects of the Coulomb

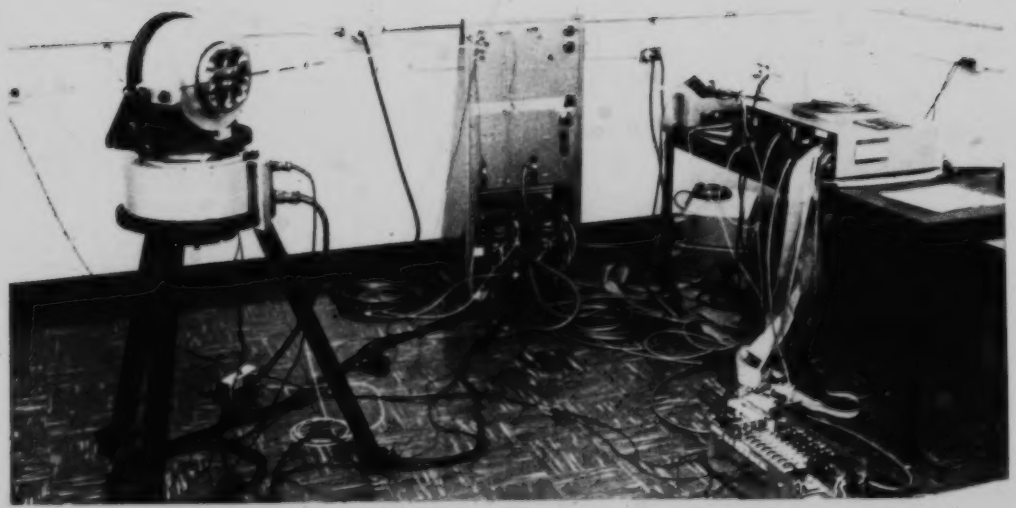


Figure 4.1: Complete test-bed.

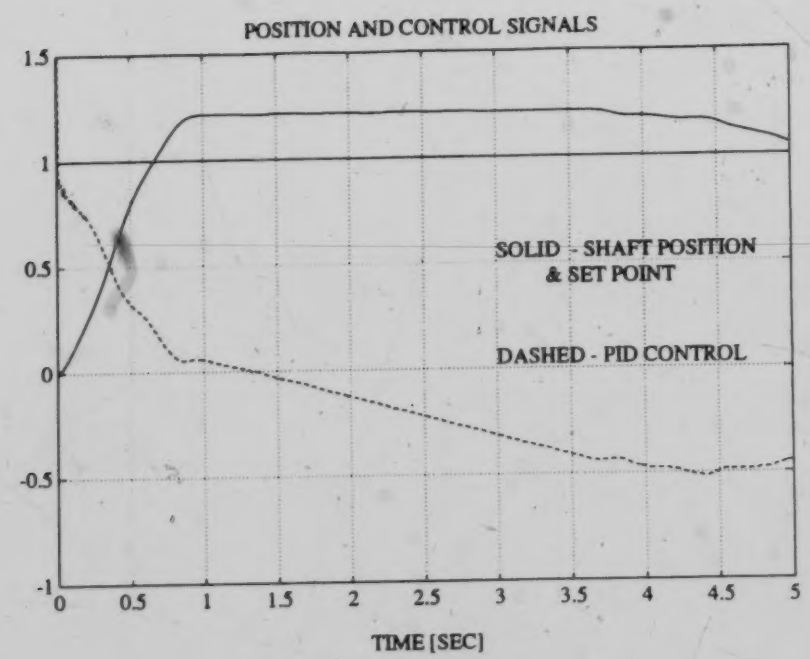


Figure 4.2: Initial step response of the base motor with a PID controller.

friction and the viscous friction are further examined in section 4.4. Experiments to determine the DD motor delays, velocity rise time, and frequency response are presented in section 4.5. A set of experiments leading to the specification of the accelerometer range is discussed in section 4.6. Double integrator attenuation of periodic signals and an analysis of discharging the double integrator are explained in section 4.7. Measurements of the flexible link modal frequencies are described in section 4.8.

4.2 Friction

The initial attempt to control the base DD motor revealed that significant friction effects exist. An initial position control closed-loop experiment with a PID controller and sampling frequency of 100 Hz is shown in Figure 4.2. The response exhibits a long steady-state error due to friction, which is eventually overcome by the integral mode of the PID controller.

There are three friction effects: stiction, Coulomb friction, and linear viscous friction. The torque T_{st} due to stiction is the minimum torque needed to start the motor motion. The torque T_{co} due to Coulomb friction is the friction torque which

opposes the motor motion when the shaft velocity is approaching zero. The torque due to viscous friction is proportional to the motor shaft velocity. Stiction and Coulomb friction are known to be functions of position, surface phenomena at microscopic scale [37], lubrication, load, temperature, etc. It is also acknowledged [89] that not enough is known about their real nature. The combined friction effects are represented by the torque $T_f(t)$, where

$$\begin{aligned} T_f(t) &= T_{st}(t) \qquad \text{for} \quad \omega = 0 \\ &= T_{co}(t) + B\omega(t) \quad \text{for} \quad \omega \neq 0 \end{aligned} \tag{4.1}$$

B is the coefficient of linear viscous friction.

If the moments of inertia and the torque-velocity characteristic $\tau_{max}(\omega)$ are known, $T_f(t)$ can be determined by subtracting the inertial torques from the torque developed by the motor. In the case of the *megatorque* DD motors the moments of inertia are not known and the torque-velocity characteristic is not available. It is known that, at a given velocity, the torque is proportional to the control voltage $u(t)$. In other words, the instantaneous torque is given by

$$\tau(\omega, u(t)) = \tau_{max}(\omega)\frac{u(t)}{u_{max}} \tag{4.2}$$

where u_{max} is the maximum control voltage. Unfortunately, $\tau_{max}(\omega)$ is not supplied. The manufacturer only claims that the motor torque has been compensated to minimize the torque ripple and that the inherently nonlinear torque-velocity characteristic of variable reluctance motors has been linearized. Values of stiction, Coulomb friction, and linear viscous friction are not provided. In the case of the *megatorque* DD motors the friction effects due to bearings and the brakes are in the range from 4% to 7% of the maximum motor torque.

The *stiction torque* can be determined experimentally when $|\tau(0, u(t))| - |T_{st}|$ becomes positive and the motor shaft starts to move. Such an experiment depends on several factors. First, it must be defined what constitutes a valid initial motion. Since the shaft position is observed as number of pulses (614400 pulses/rev), it was decided that when the shaft position changes by more than 10 pulses, a valid motion is deemed to have occurred implying that the stiction torque has been overcome. A sufficiently small control voltage (which does not cause motion) is applied to the motor at the beginning. The voltage is then increased slowly until initial motion is detected. When this happens, a PID control law is applied in order to place the motor shaft at the next position where the stiction is to be determined. The desired

position points are spaced by 100 pulses.

A program was written in C programming language which performs the entire operation in a loop, i.e. slow ramp input until the initial motion is detected followed by the position PID algorithm to reach a new position. The positional accuracy $614400/360° = 1706$ pulses/1° implies 17 measurements per one degree. Two problems occurred. First, it was not possible to stop at the exact desired positions, i.e. 100, 200, etc. Second, although the position resolution is very high, it is not currently possible to repeat the experiments from the same initial position.

It is expected that the experiments will show whether the stiction is a function of position only and whether it also depends on the direction of motion. To minimize the error due to the initial position, the motion is started from one limit switch position and was reversed in the proximity of the other limit switch position. Even with many programming improvements one experiment took over 24 hours. To ensure reliability of the results, the experiment was repeated twice. Since the torque could not be measured directly, the applied control voltage was used as a measure of the torque. Figure 4.3 shows the variation in control voltage for counterclockwise rotation from two different experiments. Note that the results are shown only for the initial 16 degrees of motion. The results for the subsequent subintervals had the same periodic character. Figure 4.4 shows the variation in the control voltage for clockwise rotation for the same path sections.

The experiment #1 is replotted in Figure 4.5 and the results for counterclockwise direction are plotted with the opposite sign. Figure 4.5 confirms symmetry of the stiction.

The periodic character of stiction corresponds to the number of pole teeth of the *megatorque* DD motors. Two explanations are possible. First, it is very likely that each pole tooth behaves like a small permanent magnet whose effects must be overcome to start the motion and therefore depends on the position. Second, the commutation linearizing algorithm may contribute to the periodical position dependency.

The above experiments show that stiction depends on the position θ and has a nonlinear character. One method of compensating for the stiction (assuming that the absolute position is known) would be to prepare a look-up table with the measured stiction so that proper values of control voltage compensation can be used. However, if the stiction is dependent on other conditions as Figure 4.3 suggests, then exact stiction compensation may not be achieved. As already mentioned the absolute shaft position is not known. Currently the look-up table stiction compensation is not

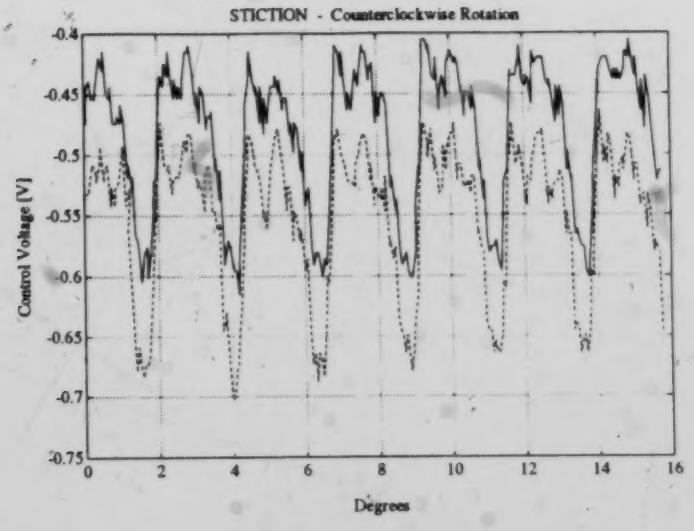


Figure 4.3: Stiction for counterclockwise rotation - two experiments.
Solid line - experiment #1
Dashed line - experiment #2

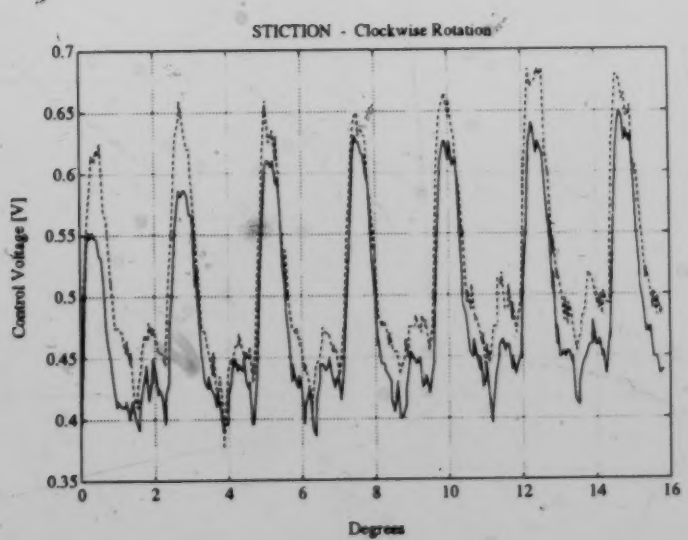


Figure 4.4: Stiction for clockwise rotation - two experiments.
Solid line - experiment #1
Dashed line - experiment #2

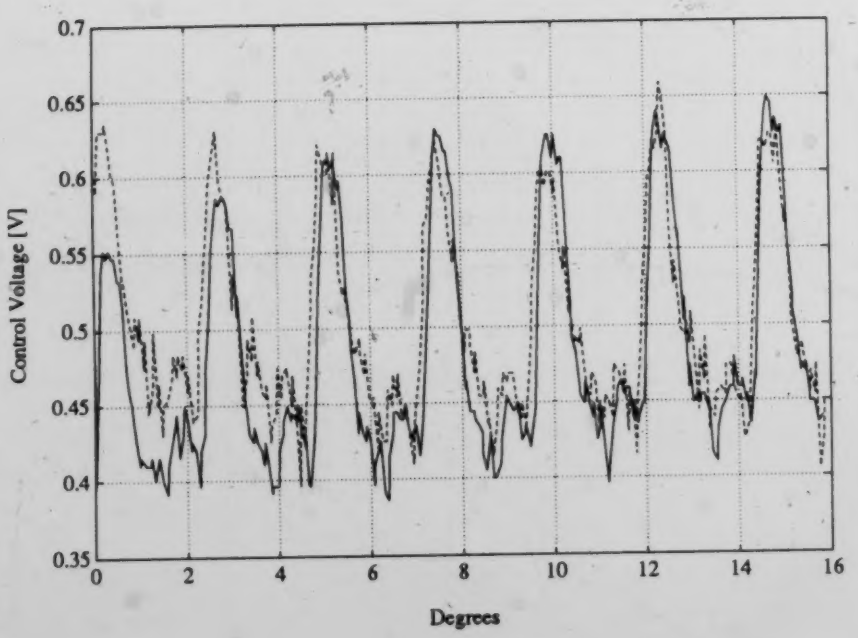


Figure 4.5: Comparison of stiction in both directions.
Solid line - clockwise rotation
Dashed line - counterclockwise rotation

Counterclockwise	0.625	0.581	0.645	0.654	0.664	0.654	0.659
Clockwise	-0.645	-0.649	-0.649	-0.640	-0.635	-0.640	-0.630

Table 4.1: Average control voltage to overcome stiction in both directions.

Counterclockwise	0.542	0.537	0.542	0.537	0.542	0.532	0.532
Clockwise	-0.552	-0.552	-0.552	-0.552	-0.542	-0.537	-0.532

Table 4.2: Average control voltage corresponding to Coulomb friction in both directions.

implemented due to the above mentioned difficulties.

It is more difficult to determine the Coulomb and linear viscous friction torques as a function of position. To find values of the Coulomb friction, the motor is run at a very slow velocity and control voltages $u(t)$ are recorded. Although such an experiment is hypothetically feasible, it is difficult to measure this control voltage accurately because the moment of inertia tends to smooth the motion and masks the variations in Coulomb friction. The exact absolute shaft position is not known.

The average value of Coulomb friction was measured by using the following experiment. A program written in C programming language allows each touch of the computer keyboard to increase or decrease the control voltage by ± 5 mV. The control voltage and the current shaft position are displayed on the monitor at the same time. When the control voltage is larger than the stiction the motor starts to move. Now, the voltage is slowly decreased and the voltage when the motor stops defines the average Coulomb friction voltage. The same approach can be used for determination of average values of the stiction, i.e. the voltage when the motor starts to move. A set of measured values is shown in Table 4.1. Comparison of the stiction values from Figures 4.3 and 4.4 and Table 4.1 shows the differences in both approaches.

The Coulomb friction plays an important role in control applications when the control torque level is in the vicinity of the Coulomb friction torque. If the control torque is compensated for the Coulomb friction by a value smaller than the real value of Coulomb friction at a particular location, then the uncompensated friction behavior will appear again but on a smaller scale. If, on the other hand, the average compensation value of the Coulomb friction is larger than the Coulomb friction at the particular point, the motor response is more active and is difficult to stop the motor at the desired position. To preserve the active motor response and avoid the possibility of a limit cycle, slightly overcompensated Coulomb friction together with actuator dead-zone are typically used in this project.

4.3 Performance of PID Controller with Different Friction Compensation Schemes

The importance of including mechanisms to compensate for the friction in any control scheme is discussed in this section. As an example, the effects of the friction compensation when used with an experimentally tuned PID controller are demonstrated. All experiments were carried out with the same controller settings and the sampling frequency of 100 Hz. It must be emphasized that only the motor shaft position and not the flexible link acceleration was controlled in this experiment. Techniques for controlling the tip acceleration are discussed in chapter 5.

In order to compensate for stiction, the control voltage is compensated by ± 0.65 V according to the polarity of the control voltage. This value is slightly larger than the average values of the stiction voltage shown in Table 4.1. The Coulomb friction compensation control voltage is set to ± 0.55 V which is slightly higher than the average value of the Coulomb friction voltages in Table 4.2

Friction Compensation.

To distinguish which compensation should be used, the control algorithm must decide whether the motor is moving or not. The *motion indication*, which is an additional tuning parameter, is expressed in terms of position pulse differences between consecutive samples. It the pulse difference is larger than the *motion indication* parameter, currently 20 pulses, it is concluded that the motor is moving.

Since the friction compensation values are slightly overcompensated, the control voltage dead-zone is used. If the control voltage is within the dead-zone range, zero torque is generated to avoid a possible limit cycle. The value of the *dead-zone* is an additional tuning parameter which for all responses presented in this section was set to ± 6 mV.

All experiments in this section deal with a step response to a step command of 1 radian.

Figure 4.6 shows the step response when no friction compensation is used. The response exhibits a long steady-state error when the control signal is smaller than the friction effects. The PID controller overcomes the steady-state error due to its integral mode.

The response in Figure 4.7 was obtained only when the Coulomb friction compensation is used. The spikes in the control signal are due to the Coulomb friction compensation and appear when the algorithm perceives that the motor is moving and the control voltage is larger than the control voltage dead-zone. Notice that in the

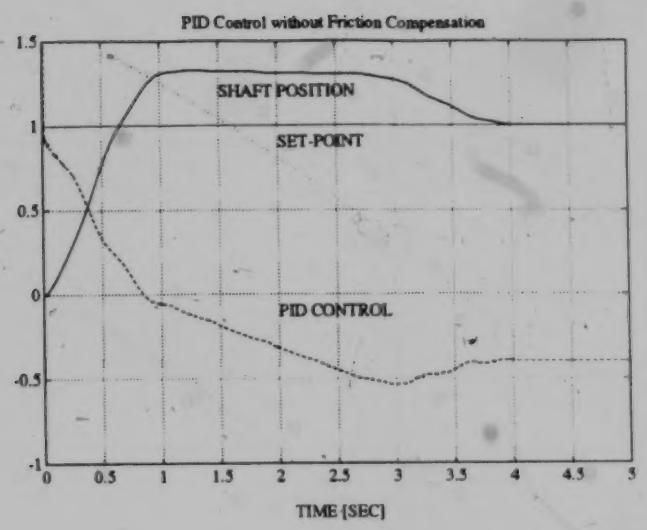


Figure 4.6: Step response with uncompensated PID controller.

second half of Figure 4.7 a small steady-state error still exists but the motor is not moving and the Coulomb friction compensation is not activated. The small steady-state error is also indicated by increasing the control signal towards the end of the response.

If only the stiction compensation is used with the same settings of the PID controller, the step response shown in Figure 4.8 is obtained. The train of spikes in the control voltage corresponds to the stiction compensation which occurs whenever the motor stops and the control signal is larger than the control voltage dead-zone. However, at the instant when the motor starts to move again the stiction compensation stops and the motor is driven by the torque due to the control voltage only. If the control voltage is smaller than the missing Coulomb friction then the motor stops and the stiction compensation is activated again. This results in a slow motion that frequently stops. The response is rather sluggish with very pronounced pulses due to the stiction compensation.

Finally, Figure 4.9 shows the step response when both friction compensations are used. The steady-state level is reached in about 1.2 sec.

Generally, the responses with friction compensation show significant improvement over the uncompensated response. However, the spike character of the friction introduces pulse excitation which is not desirable when controlling flexible structures.

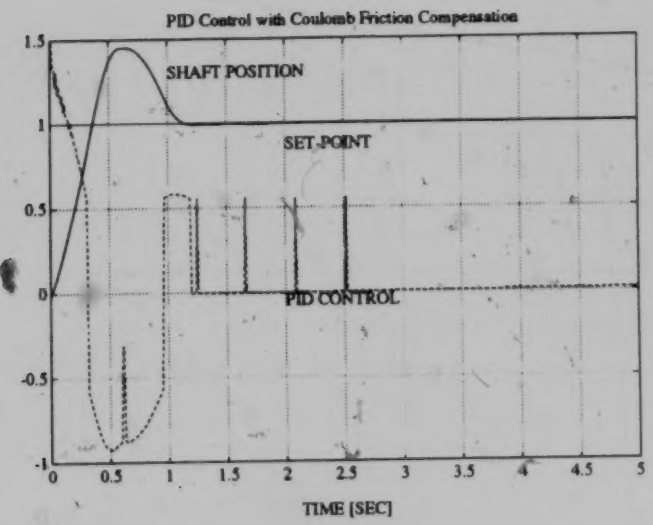


Figure 4.7: Step response of a PID controller with Coulomb friction compensation.

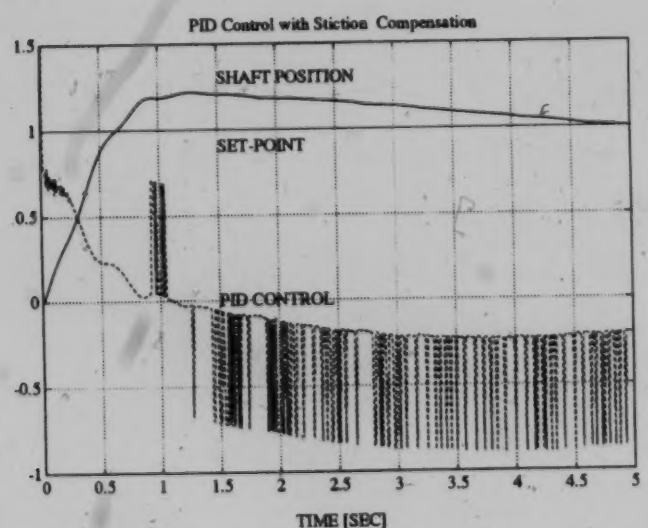


Figure 4.8: Step response of a PID controller with stiction compensation.

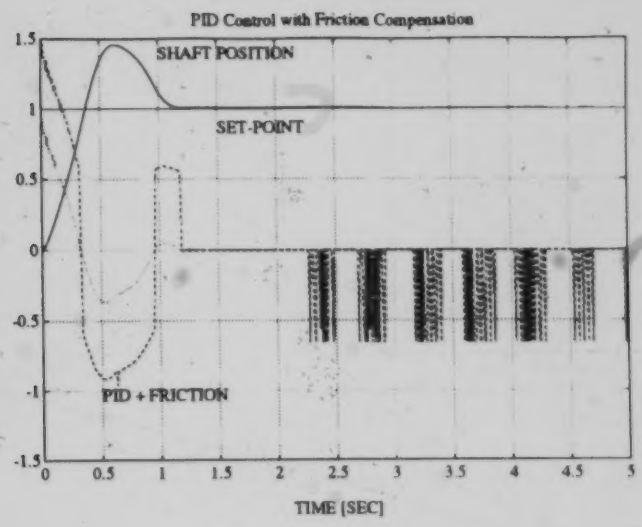


Figure 4.9: Step response of a PID controller with stiction and Coulomb friction compensation.

4.4 Combined Coulomb and Viscous Frictions

Separation of the Coulomb and viscous friction effects is impossible because both occur when $\omega \neq 0$. Nevertheless, it is important to know whether their combined effect is linear or nonlinear. If the motor torque τ and the moments of inertia are known, then the combined effects of Coulomb friction and viscous friction can be determined from the following equation

$$B\omega + T_{co} = \tau - J\frac{d\omega}{dt} \tag{4.3}$$

However this is not possible since the motor torque and moments of inertia are not known. If the motor torque τ is set to zero then the above equation can be written as

$$0 = J\frac{d\omega}{dt} + B\omega + T_{co} \tag{4.4}$$

which leads to

$$-J\frac{\epsilon}{\omega} = B + \frac{T_{co}}{\omega} \tag{4.5}$$

where $\epsilon = d\omega/dt$ is the angular acceleration. J is assumed to be constant and without loss of generality it can be set equal to 1.

If B and T_{co} were constants, then $-\epsilon/\omega$ should be a monotonic decreasing function of ω. To verify whether this is true or not, the following experiment was

carried out. After the motor reaches its maximum speed of 10 rad/sec, the control voltage is changed to zero instantaneously and values of position and velocity are sampled at 500 Hz.

To plot $-\epsilon/\omega$, it is necessary to obtain the motor shaft angular acceleration ϵ from the velocity measurements because the angular acceleration of the motor cannot be measured directly. The neighborhoods of the velocity boundaries $\omega = 0$ and $\omega = \omega_{max}$ are discarded from the velocity measurements to avoid division by zero in equation 4.5 and to diminish effects of irregularities in velocity measurements occurring at $\omega = \omega_{max}$ when the control voltage is lowered to zero. The angular acceleration is obtained from the measured velocity by five-point numerical differentiation followed by an 8-th order Butterworth low-pass filter (LPF) to eliminate the amplified noise due to differentiation.

The analog velocity signal comes from 8-bit DAC (while Lab Master is a 12-bit ADC) and is relatively coarse because of significant quantization effects. The quantization effects of the 8 bit DAC can be seen as a noise and therefore the measured velocity is also filtered by the same 8-th order Butterworth LPF. The data was processed off-line using MATLAB. The velocity and acceleration are plotted in Figures 4.10A and 4.10B, respectively. The $-\epsilon/\omega$ as a function of ω is shown in Figure 4.10C.

To verify the validity of this procedure, another method was used to process the same experimental results. Position measurements from the QDB were used to obtain ω and ϵ. The velocity and acceleration are plotted in Figures 4.11A and 4.11B, respectively. The $-\epsilon/\omega$ as a function of ω is plotted in Figure 4.11C.

The two graphs of $-\epsilon/\omega$ in Figures 4.10C and 4.11C exhibit the same irregularity in the sense that in both cases the graph decreases monotonically up to ≈ 8 rad/sec and then starts to increase. This non-monotonic behavior can be attributed to combined effects of T_{co} and B being nonlinear. It is also possible that the description of the motor dynamics by a second order differential equation may be too simple; a higher order model is called for. This may have contributed to the observations in Figures 4.10C and 4.11C.

No matter where this phenomenon comes from, it is obvious that it represents a nonlinear effect and the control algorithm must be robust enough to deal with it.

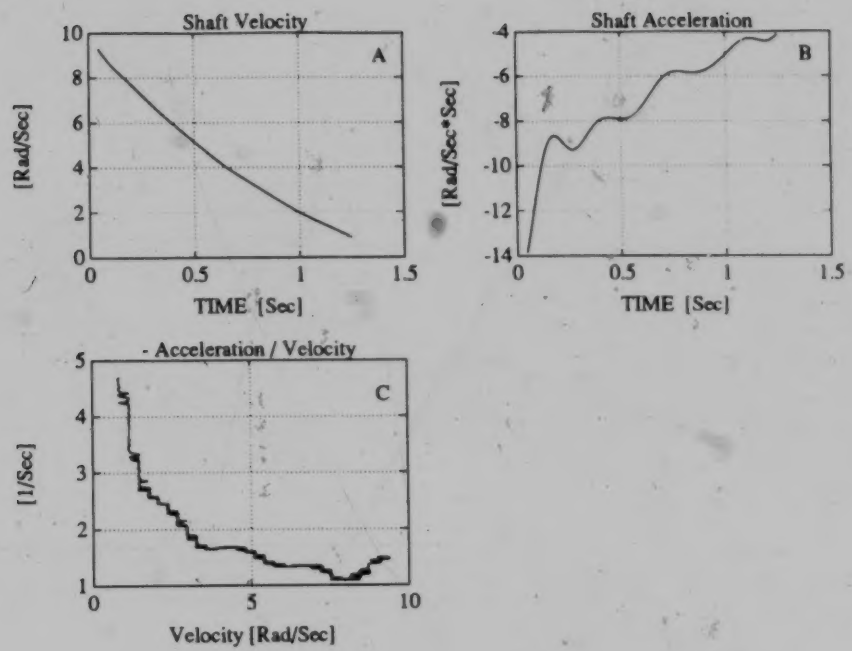


Figure 4.10: Characteristics based on measured velocity.

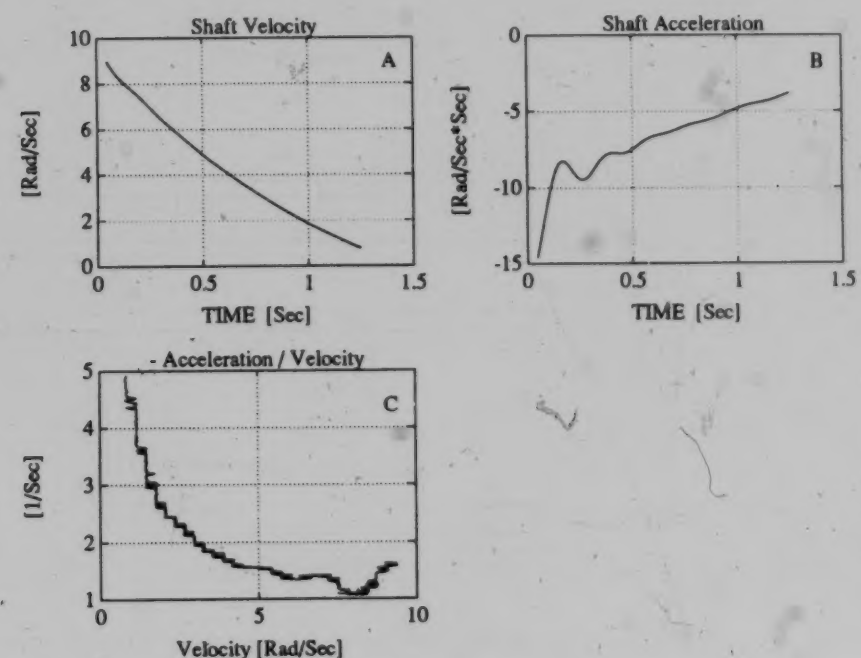


Figure 4.11: Characteristics based on measured position.

4.5 DD Motor Dynamics

The measurements of some of the parameters and the frequency response of the DD motors are described in this section.

4.5.1 Motor Delays and Rise Time $t_{\omega_{max}}$

When a control voltage is applied to the DD motor, there is a delay before the motor responds by a change in its position and velocity. This delay is due to the combined effects of the A/D converter, digital signal processor (DSP), and the motor amplifier hardware. Knowledge of this time delay plays a significant role in the control algorithm discussed in the next chapter. Another parameter of interest is the rise time in velocity $t_{\omega_{max}}$. The motor delay and the rise time are obtained from open-loop measurements.

The position delay $t_{\theta_{up}}$ and the velocity delay $t_{\omega_{up}}$ are determined from an open-loop step response. The shaft position is obtained from QDB and the measured velocity from the DD motor analog subsystem. All delays are measured for different values of the applied control voltage. Note that these measurements were done without the friction compensation because the applied control voltage was always above the friction compensation voltage.

The rise time $t_{\omega_{max}}$ was defined as the time needed for changing the velocity ω from $\omega = 0$ to $\omega = \omega_{max} = 10$ rad/sec. This definition is used because the motor amplifier is working in the torque mode in which it delivers torque proportional to the control voltage. The maximum velocity that can be attained is 10 rad/sec. Transition time from $\omega = 0$ to $\omega = \omega_{max}$ depends on the applied control voltage. Table 4.3 shows the measured results for the base motor. A sampling frequency of 2 kHz was used for all measurements in the table.

4.5.2 Open-Loop Frequency Response

The purpose of this section is to obtain the frequency response (amplitude and phase) of the motor velocity so that the frequency characteristics and bandwidth can be determined.

An experiment, controlled by a C language program, was carried out to determine, on line, the amplitude and the phase of the motor velocity at specified frequencies f_i $(i = 1, 2, \ldots)$. The program generates the sinusoidal control voltage and samples the motor velocity at sampling frequencies f_{s_i} $(i = 1, 2, \ldots)$. The

u_c [V]	$t_{\theta_{up}}$ [msec]	$t_{\omega_{up}}$ [msec]	$t_{\omega_{max}}$ [msec]
10.0	3	2	50
9.0	3	2	53
8.0	4	4	60
7.0	3	3	66
6.0	3	3	80
5.0	4	4	102
4.0	4	3	130
3.0	4	4	180
2.0	4	4	365
1.5	4	3	520
1.0	5	4	900
0.8	5	4	∞

Table 4.3: Measured delays and rise time.

quality of the sinusoidal control voltage and the measurement accuracy depends on the sampling frequency f_{s_i} which is controlled by the hardware counter timer 9513A. To preserve the same measurement accuracy for different values of f_i, the sampling frequency f_{s_i} is chosen to be

$$f_{s_i} = f_i * SIN\_POINTS \quad \text{for} \quad i = 1, 2, \ldots \tag{4.6}$$

where the variable SIN_POINTS defines the number of samples over one complete period.

It is necessary to stop and reinitialize the hardware counter timer 9513A when the measurements and evaluation of the amplitude and phase of the velocity are completed for a particular frequency f_i. Note that the points of the sinusoidal profile are calculated only once and stored in an array from which they are retrieved in a circular fashion with each sampling frequency f_{s_i}.

To eliminate effects of the transient, occurring when a new frequency is applied, the user touches a key on the computer keyboard when the motor motion exhibits regular sinusoidal behavior.

Measurements of amplitude. The output amplitude is determined by the following equation

$$A_{out} = \frac{1}{4}\left(\text{abs}(A_1^+ - A_1^-) + \text{abs}(A_2^+ - A_2^-)\right) * VEL\_per\_VOLT \tag{4.7}$$

where the constant VEL_per_VOLT is the velocity output conversion factor equal to

1.66. The positive peak values A_1^+ and A_2^+ and the negative peak values A_1^- and A_2^- are determined when the system undergoes 2 complete periods at frequency f_i after the steady-state response is achieved.

The amplitude response, as a ratio A_{out}/A_{in} is presented in Table 4.4. A_{in} is the amplitude of the input control voltage. When the motor torque is generated by the sinusoidal control voltage with an amplitude greater than 2 Volts, the velocity, if allowed, would reach values greater than 10 rad/sec but it is clipped at 10 rad/sec by the motor amplifier. Therefore, for larger control voltages and smaller frequencies the amplitude characteristic does not change as can be seen in Table 4.4. This effect is quite noticeable for the control voltage $u_c = 4$ Volts and frequencies $f \in < 0.5, 1 >$ Hz. The amplitude response, as presented in Table 4.4, was obtained as a ratio A_{out}/A_{in}. The velocity saturation also explains different values of the amplitude characteristic when different $u_c(t) = A_{in} \sin \omega t$ were applied.

The values in Table 4.4 were obtained as averages from several runs and the results were consistent. The calculation of A_{out} according to equation 4.7 accommodated a distorting effect of a slow superimposed rotation of the motor which occurred randomly at different measured frequencies f_i.

Measurement of phase. Let t_{max} be the instant when the peak A_1^+ occurs and measured as the number of timer interrupts since the initial point of the sinusoidal control voltage after the user acknowledges the steady response for any f_i. The phase shift for the particular frequency f_i is calculated as follows

$$\phi(f_i) = \left(t_{max} - \frac{SIN\_POINTS}{4}\right) \frac{360^\circ}{SIN\_POINTS} \tag{4.8}$$

The measured values of the phase shift are shown in Table 4.5. Special care should be taken in determining t_{max}. The 8-bit built-in DAC of the *megatorque* motor analog interface sometimes generates spikes which do not correspond to the actual velocity. The occurrence of these spikes is not predictable. Consequently, several runs of the experiments were made at each frequency f_i. The lowest values of t_{max} was used in equation 4.8.

According to Motornetics [76], the manufacturer of the *megatorque* motors, the velocity bandwidth should be close to 10 Hz for the motor without any load. Since the base motor has an inertial load in the form of the shoulder motor and the steel bracket holding the shoulder motor at its location, it can be concluded that the entire moment of inertia is larger by one or two orders of magnitude, than the inertia of the base motor rotor. This explains why in the Table 4.4, the velocity bandwidth

f[Hz]	0.5	0.6	0.7	0.8	0.9	1.0	2.0	3.0	4.0	5.0
	Normalized Velocity Amplitude A_{out}/A_{in} [-]									
$u_c = 1V$	4.58	3.77	3.23	2.94	2.63	2.50	1.41	1.04	0.79	0.66
$u_c = 2V$	4.85	4.66	4.04	3.58	3.12	2.90	1.45	1.02	0.79	0.67
$u_c = 3V$	3.23	3.23	3.23	3.23	3.22	3.19	1.51	1.04	0.77	0.64
$u_c = 4V$	2.42	2.42	2.42	2.42	2.42	2.42	1.73	1.07	0.82	0.67

Table 4.4: Amplitude response to a sinusoidal input.

f[Hz]	0.5	0.6	0.7	0.8	0.9	1.0	2.0	3.0	4.0	5.0
	Phase Shift [Degrees]									
$u_c = 1V$	41	44	45	55	57	61	63	63	98	77
$u_c = 2V$	45	57	63	65	69	74	74	84	99	91
$u_c = 3V$	19	23	28	54	56	79	85	92	97	84
$u_c = 4V$	27	29	32	64	32	43	84	90	99	109

Table 4.5: Phase shift of the response to a sinusoidal input.

for $u_c = 1$ V is about 0.7 Hz.

One other potential problem should be pointed out. It is possible that at high sampling frequencies some samples may be missed. To find out whether this happened, all samples at the highest frequency are stored in a file with one additional variable. This variable is always incremented whenever an interrupt due to the 9513A timer interrupt occurs, i.e. the increment of this variable is a part of the timer interrupt service routine. When the experiment is finished, a MATLAB program is called which not only displays results from the highest frequency measurements but also checks for skipped samples.

4.6 Determination of the Accelerometer Range

The role of the accelerometer in the test-bed is to measure the acceleration of the tip of the flexible link. The determination of the appropriate acceleration range for the accelerometer depends on several factors. An accelerometer with too large a range will have poor resolution. On the other hand, the accelerometer with too small a range will reach its saturation limit and face the possibility of a permanent damage.

Two sets of open-loop experiments and a closed-loop response test with a PID controller were carried out on the manipulator with two accelerometers obtained on loan.

Let us assume that the link is rigid and the motor torque has sinusoidal shape with a known frequency. If the amplitude of motion of the tip of the flexible link is

f [Hz]	$2A$ [m]	a_{max} [g]	$a_{max_{MEASURED}}$ [g]
3	0.22	3.98	5.4
4	0.12	3.86	5.2
6	0.02	1.45	2.4
8	0.02	2.57	3.5

Table 4.6: Acceleration from open-loop sinusoidal experiments.

small, then an accelerometer located at this tip measures the tangential acceleration. The tangential acceleration $a(t)$ is approximately given by the following equation

$$a(t) = l\epsilon(t) = l\frac{d^2\theta(t)}{dt^2} = l\frac{d^2}{dt^2}A\sin(\omega t) = -lA\omega^2\sin(\omega t) \qquad (4.9)$$

where $\epsilon(t)$ is motor shaft angular acceleration, l is the link length, ω is the angular frequency of the link sinusoidal motion, and A is the amplitude of the link tip motion. The maximum amplitude of the tangential acceleration expressed in g's (gravitational acceleration) is determined for $\sin(\omega t) = \pm 1$

$$|a(t)_{max}| = \frac{lA\omega^2}{9.81} = \frac{lA(2\pi f)^2}{9.81} \qquad (4.10)$$

The experiments were carried out using two Brüel & Kjaer accelerometers one with $79.6mV/g$ resolution and another with $20.8mV/g$ resolution. The results for four frequencies are shown in Table 4.6. The length of the link for this experiment was set to 1m.

The measured acceleration is composed of acceleration due to rigid and vibrational motions. Consequently the values recorded in Table 4.6 are larger then the values of the calculated accelerations according to equation 4.10.

The maximum possible acceleration due to the motor motion occurs when a step input voltage is applied to the motor in open-loop control configuration. Table 4.7 shows the maximum angular accelerations of the shaft and corresponding tangential tip accelerations calculated for the link length l = 1.5 m according to $a_{max} = l\epsilon_{max_{up}}/9.81$. The angular accelerations were obtained by five-point numerical differentiation followed by low-pass filtering.

The angular deceleration obtained when the step input control voltage was lowered to zero exhibits values around 10 rad/sec^2 which corresponds to a tip tangential deceleration of 1.52g. Note, that these results could not be verified by direct accelerometer measurements because the motors were rotating freely and the cables from the accelerometer to the signal conditioning unit had to be disconnected.

u_c	$\epsilon_{max_{op}}$ [rad/sec^2]	a_{max} [g]
10	203	31.0
5	101	15.4
3	50	7.6
2	28	4.2
1	14	2.1
.8	5	0.7

Table 4.7: Maximum accelerations from open-loop step measurements.

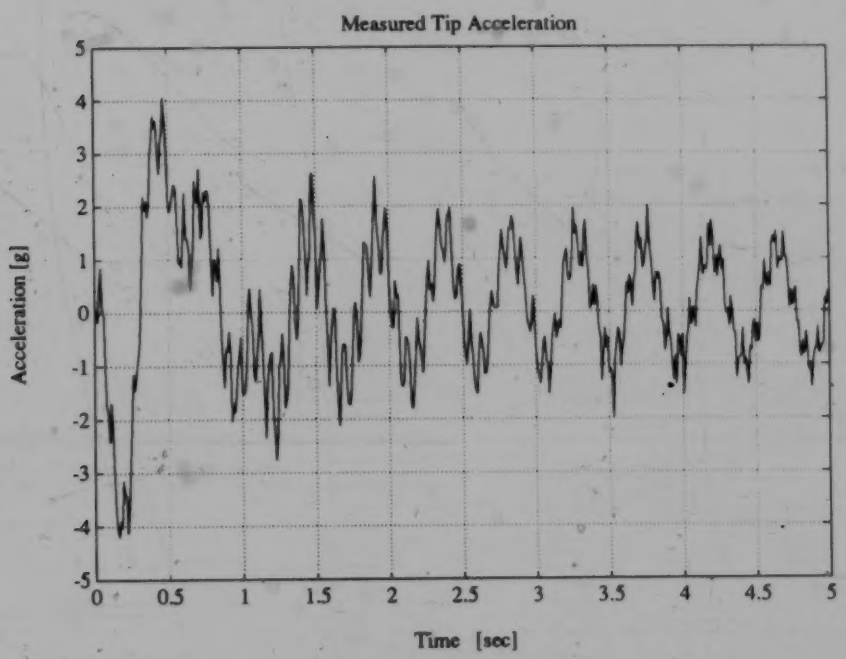


Figure 4.12: Measured acceleration from a closed-loop experiment.

The acceleration measurements from a closed-loop control experiment were carried out with a PID controller. The details of the experiment are discussed in section 4.3. The measured accelerations from this experiment are shown in Figure 4.12.

Although, to be on the safe side, the worst case scenario coming from the open-loop step measurements shown in Table 4.7 should be used to determine the required accelerometer range, the following mitigating factors favor a lower range:

- When the maximum shaft acceleration occurs at the beginning of the step response, the measured acceleration is smaller because the flexible link tip does not follow the rigid motion instantaneously due to its flexibility.

- When a good closed-loop control algorithm is used for positional control, the rigid shaft acceleration will be smaller than the worst case step input open-loop accelerations.

Taking the above factors as well as cost into account, an accelerometer with $\pm$ 50g range has been selected together with a signal conditioning unit which has an adjustable gain in the range $< 0.5, 5 >$. Currently the adjustable gain is set to 5.

4.7 Factors Affecting the Double Integrator Performance

The purpose of the double integrator was explained in section 3.7. In using the double integrator, certain factors must be considered to obtain reliable and accurate measurements. These are discussed in this section.

The double integrator was first tested with a pulse of known amplitude and duration generated by the analog interface of the host 386 computer. The performance was found to be satisfactory.

Since the output of the integrator is limited by $\pm$10 Volts, discharging switches controlled by the host 386 computer are activated whenever the output voltage approaches $\pm$10 Volts. Tests were performed to determine the speed and quality of discharge. The discharge switches guarantee good (1%) discharging up to a sampling frequency of 100 Hz as anticipated in chapter 3.

Attenuation Effects. Attenuation of periodic signals is inherent in integration. If the frequency of the input signal is ω the output signal will be attenuated by $1/\omega$. For example, if the input signal frequency varies from 3 to 4 Hz, ω varies from 18.86 to 25.13. If the acceleration signal corresponds to $\pm$10 V, the corresponding velocity signal will yield a voltage of about 500 mV and the position signal will correspond to only about 25 mV. This presents a significant measurement resolution problem. This deficiency can be overcome by selectively increasing the gain of each integrator if the frequency of the vibrations was narrow. However, in practice, assumption of a narrow frequency range may not be realistic and it is difficult to achieve a reasonable constant gain compensation for the entire frequency range. While gain compensation improves measurement accuracy, it brings the integrator much faster to saturation when a low frequency input appears. In the sequel velocity and position signals $v(t)$ and $s(t)$ are multiplied by gains k_v and k_s by the respective integrators.

Discharging Effects. When a comparison was made between signals measured

from the double integrator and values obtained via off-line numerical integration of the accelerometer signal, it became obvious that the discharging process needs to be taken into account. Part of the double integration process must be carried out numerically so that correct information is available, irrespective of whether discharging occurred or not. The program must compensate for errors due to interrupted hardware integration during the sampling period when discharging is activated. Modification of the entire algorithm is required for the case when incomplete discharging occurs because of short sampling period. Programming factors are included in the following discussion.

In the sequel barred variables denote signals measured directly from the integrators. Let the time of commencement of integration be t_o, let the activation of the discharging process occur at $t = t_1$ and let T denote the sampling period.

Case I: Ideal case. No saturation occurs.

For an acceleration signal $a(t)$, the velocity is obtained as

$$v(t) = VEL + k_v \int_{t_o}^{t} a(\tau)d\tau = VEL + \bar{v}(t) \tag{4.11}$$

where

$$VEL := v(t_o) \tag{4.12}$$

where the variable VEL is in the computer memory and holds the value of the velocity $v(t)$ when the double integrator starts to integrate. The assignment sign ":=" is used to express that a variable is assigned a new value in the programming sense because use of the "=" would be mathematically incorrect.

Similarly, the position is given by

$$s(t) = POS + k_s \int_{t_o}^{t_1} \bar{v}(\tau)d\tau = POS + \bar{s}(t) \tag{4.13}$$

where

$$POS := s(t_o) \tag{4.14}$$

and where the variable POS holds the position value at the time when the integration commences.

Case II: Complete discharge occurs in one sampling period.

Suppose the output of one of the integrators reaches the saturation level at time t_1. The host computer triggers the discharging process in both integrators at time t_1 and also disconnects the input signals from the integrators. The duration of the discharging process is one sampling period from t_1 to $t_1 + T$. Figure 4.13 shows

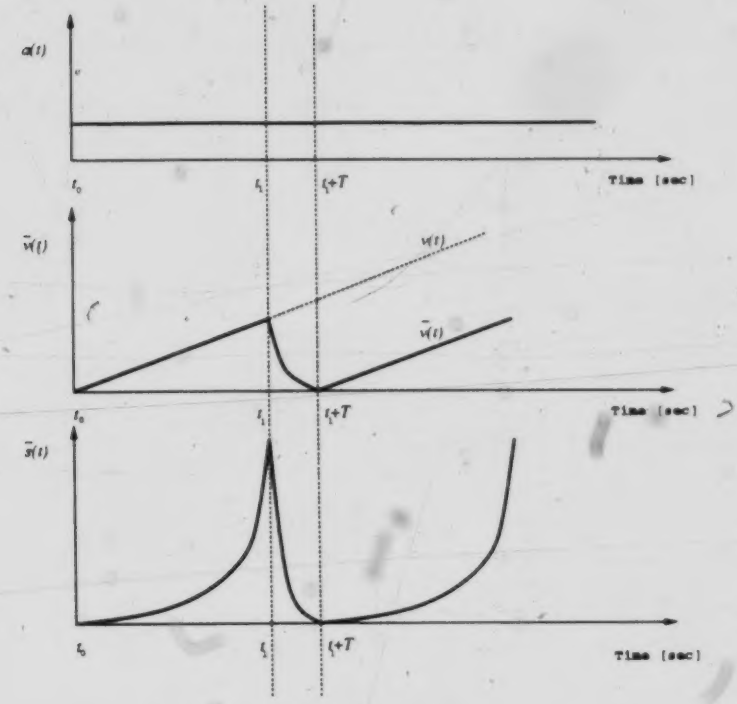


Figure 4.13: Complete discharge - an example.

a contrived example when the acceleration is constant and initial conditions of $\bar{v}(t)$ and $\bar{s}(t)$ are zero. Then the velocity measurement $v(t)$ at the time $t > t_1 + T$ can be expressed as a sum of three integrals

$$v(t) = VEL + k_v \int_{t_o}^{t_1} a(\tau)d\tau + k_v \int_{t_1}^{t_1+T} a(\tau)d\tau + k_v \int_{t_1+T}^{t} a(\tau)d\tau \qquad (4.15)$$

Since during the interval from t_1 to $t_1 + T$ no integration will be performed by the integrators, numerical integration is used. The following two equations describe how the variable VEL must be updated at the times $t = t_1$ and $t = t_1 + T$

$$VEL := v(t_1) = VEL + k_v \int_{t_o}^{t_1} a(\tau)d\tau \qquad (4.16)$$

$$VEL := v(t_1 + T) \cong VEL + k_v \frac{T}{2}(a(t_1) + a(t_1 + T)) \qquad (4.17)$$

Then the velocity $v(t)$ is given as

$$v(t) = VEL + k_v \int_{t_1+T}^{t} a(\tau)d\tau = VEL + \bar{v}(t) \qquad \text{for } t \geq t_1 + T \qquad (4.18)$$

Following a similar procedure for the position variable the following equations are obtained

$$s(t) = POS + k_s \int_{t_o}^{t_1} \bar{v}(\tau)d\tau + k_s \int_{t_1}^{t_1+T} v(\tau)d\tau + k_s \int_{t_1+T}^{t} v(\tau)d\tau \qquad (4.19)$$

$$POS := s(t_1) = POS + k_s \int_{t_o}^{t_1} \bar{v}(\tau)d\tau \qquad (4.20)$$

$$POS := s(t_1+T) = POS + k_s \int_{t_1}^{t_1+T} v(\tau)d\tau \cong POS + k_s \frac{T}{2}(v(t_1) + v(t_1+T)) \qquad (4.21)$$

$$s(t) = POS + k_s v(t_1+T)(t-(t_1+T)) + k_s \int_{t_1+T}^{t} \bar{v}(\tau)d\tau \qquad \text{for} \quad t \geq t_1+T \qquad (4.22)$$

Note that the actual velocity $v(t)$ is used in equation 4.19 and not the measured velocity $\bar{v}(t)$.

Case III: Incomplete discharge.

Sometimes the sampling period is such that there is not sufficient time for complete discharge of the integrators to take place. Figure 4.14 shows this situation in the case when acceleration is constant and initial conditions of $\bar{v}(t)$ and $\bar{s}(t)$ are zero. Note that in this case $\bar{v}(t_1 + T) \neq 0$ and $\bar{s}(t_1 + T) \neq 0$.

The algorithm for this case is described below:

$$VEL := v(t_o) \qquad (4.23)$$

$$v(t) = VEL + k_v \int_{t_o}^{t} a(\tau)d\tau \qquad \text{for} \qquad t \leq t_1 \qquad (4.24)$$

$$VEL := v(t_1) = VEL + k_v \int_{t_o}^{t_1} a(\tau)d\tau \qquad (4.25)$$

$$v(t_1+T) \cong VEL + k_v \frac{T}{2}(a(t_1) + a(t_1+T)) = DIFFVEL + \bar{v}(t_1+T) \qquad (4.26)$$

$$VEL := DIFFVEL = v(t_1+T) - \bar{v}(t_1+T) \qquad (4.27)$$

$$v(t) = VEL + k_v \int_{t_1+T}^{t} a(\tau)d\tau \qquad \text{for} \qquad t > t_1+T \qquad (4.28)$$

where the variable $DIFFVEL$ is the difference between the real velocity $v(t_1+T)$ and the velocity $\bar{v}(t_1 + T)$ from the double integrator due to the incomplete discharging

Similarly, to obtain the correct position when incomplete discharging occurs the position measuring algorithm must be modified as follows

$$POS := s(t_o) \qquad (4.29)$$

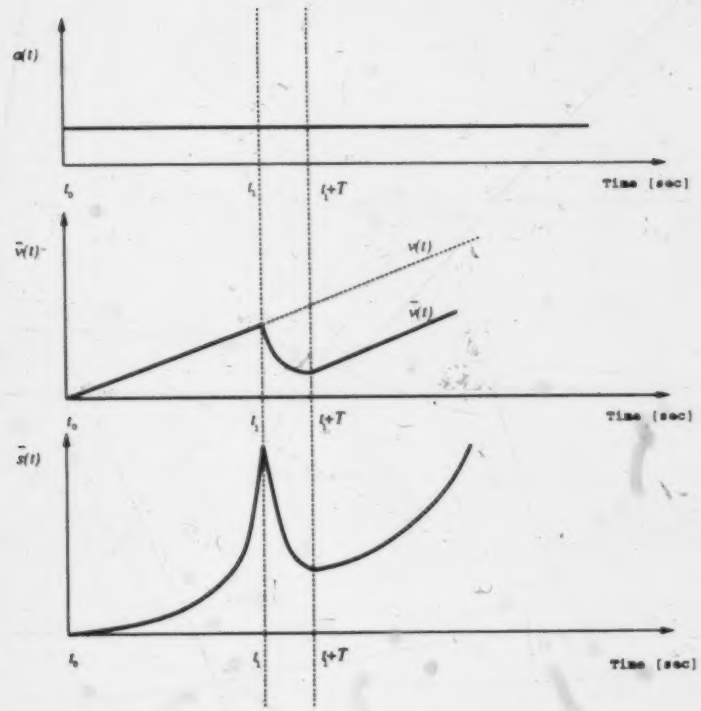


Figure 4.14: Incomplete discharge - an example.

$$s(t) = POS + k_s \int_{t_o}^{t_1} \bar{v}(\tau)d\tau \qquad \text{for} \qquad t \leq t_1 \tag{4.30}$$

$$POS := s(t_1) = POS + k_s \int_{t_o}^{t_1} \bar{v}(\tau)d\tau \tag{4.31}$$

$$s(t_1+T) = POS + k_s \int_{t_1}^{t_1+T} v(\tau)d\tau \cong POS + k_s \frac{T}{2}(v(t_1) + v(t_1+T)) \tag{4.32}$$

$$s(t_1+T) = DIFFPOS + \bar{s}(t_1+T) \tag{4.33}$$

$$POS := DIFFPOS = s(t_1+T) - \bar{s}(t_1+T) \tag{4.34}$$

$$s(t) = POS + k_s DIFFVEL(t-(t_1+T)) + k_s \int_{t_1+T}^{t} \bar{v}(\tau)d\tau \qquad \text{for} \qquad t > t_1+T \tag{4.35}$$

4.8 Measurements of Flexible Link Frequencies

Knowledge of flexible link frequencies is important for the development of the flexible link model used either for simulations and model-based non-adaptive control algorithms or for good initialization of on-line identification procedures which are part of adaptive control algorithms. The measured frequencies of the flexible link from the clamped-free and closed-loop configuration experiments are described in this section.

4.8.1 Clamped-Free Modal Frequencies

Modal frequencies are measured during free motion, which is characterized by zero external forces and zero moments acting on the flexible link. These can be generated by introducing non-zero initial conditions.

The following method was used to excite the modal frequencies of the flexible link. A periodic torque of short duration is applied to the motor shaft. The motor shaft starts to move and the flexible link starts to vibrate. The motor motion continues until its friction torque exceeds the vibrational torque transmitted to the motor from the flexible link. The link continues to vibrate even after the motor motion stops. Acceleration is recorded after the motor stops moving.

The measured accelerations and their power spectral densities are shown in Figures 4.15 and 4.16 for two different initial applied torques. Figure 4.15 shows the situation when the first modal frequency dominates and the third modal frequency is not excited. Figure 4.16 shows the case when the second modal frequency was excited more than the first modal frequency. It is worth noting that although the

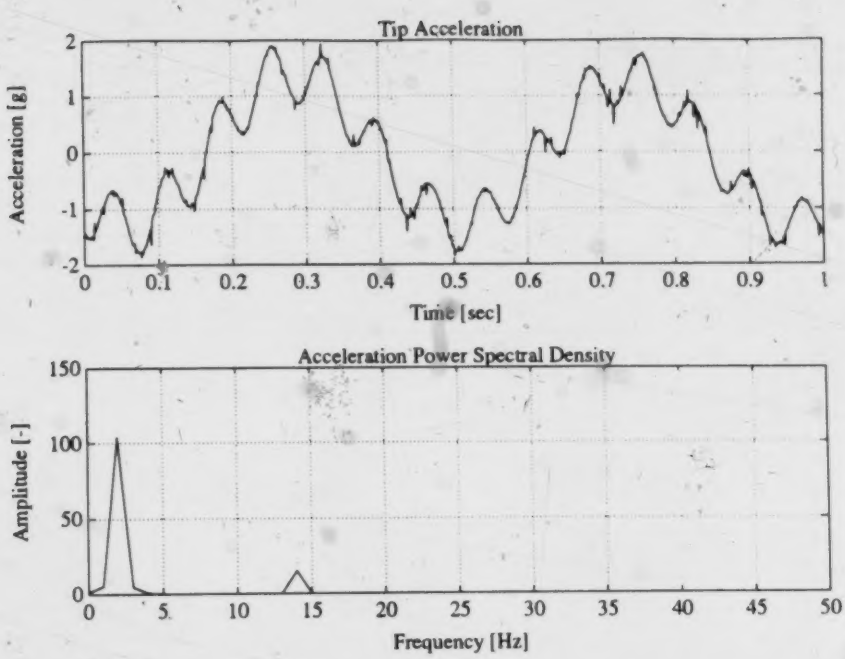


Figure 4.15: Acceleration and its power spectral density when only 2 modal frequencies are excited.

flexible link was excited in different way, the FFT showed the presence of the same modal frequencies: 2, 14, and 39 Hz. Only amplitudes were different. Neither of the experiments managed to excite the third modal frequency significantly.

The modal frequencies determined by the above method are consistently lower than the theoretical clamped-free frequencies shown in Table 2.3. This discrepancy is due to the mass of the accelerometer located at the tip of the flexible link.

4.8.2 Link Frequencies in Closed-loop Configuration

Since the flexible link responds to the applied torque, the frequency spectrum of measured accelerations are influenced not only by the link modal frequencies but also by the frequency spectrum of the applied torque. An experiment with a PID controller with full friction compensation is carried out (the details are described in section 4.3). Control voltages and accelerations were measured and analyzed by FFT. For reasons stated earlier (section 4.2), the control voltage is used as a measure of the applied torque. Figure 4.17 shows the applied control voltage and its power spectrum density while Figure 4.18 shows the measured acceleration and its power spectrum density. The significant numerical values of the spectral densities are shown in Tables 4.8 and

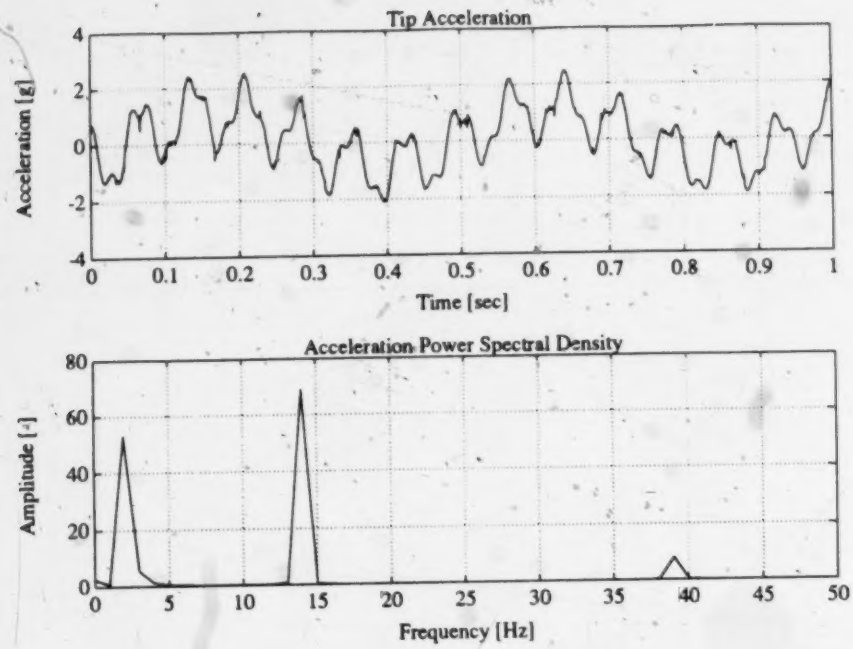


Figure 4.16: Acceleration and its power spectral density when the first 3 modal frequencies are excited.

Frequency [Hz]	0.2	0.8	2.0	2.8	3.8
Amplitude [-]	9.3	14.9	4.2	1.1	1.1

Table 4.8: Power spectral density of the control input signal.

4.9. The second and third modal frequency did not change noticeably while the first modal frequency shows a change from 2 Hz to 2.2 Hz.

These measurements also demonstrate a difficulty which arises when a closed-loop control is applied. First, the link is moving under compensated friction and its behavior approaches dynamics of a pin-free beam, with the pin end having nonzero moment of inertia, driven by a non-zero motor torque. During the final phase the applied control torque falls into the dead-zone at several instants of time, which means that no friction compensation is used and the flexible link is experiencing more and more the clamped-free boundary conditions. This means that the modal

Frequency [Hz]	0.0	1.0	2.2	14.0	39.0
Amplitude [-]	6.3	48.0	77.0	32.0	2.0

Table 4.9: Acceleration power spectral densities.

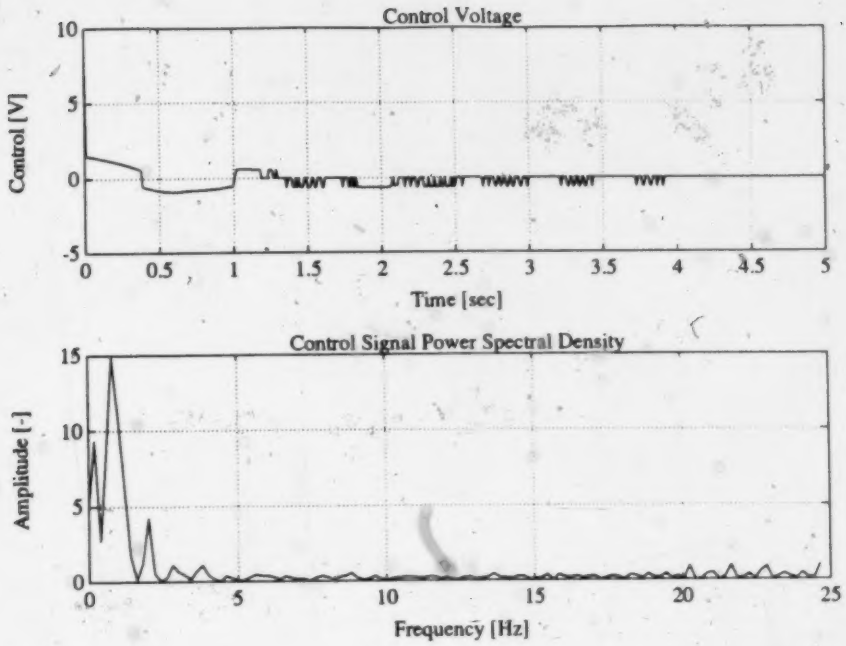


Figure 4.17: Applied control voltage and its power spectral density during a closed-loop experiment with a PID controller.

frequencies change during the controlled motion and present a significant challenge to identification and control.

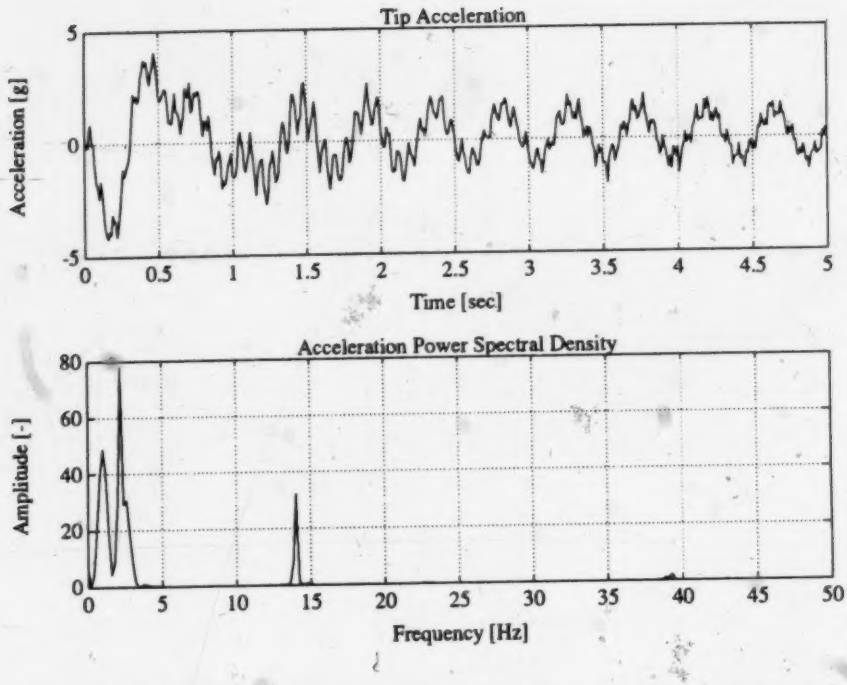


Figure 4.18: Flexible link acceleration and its frequency spectrum during a closed-loop experiment with a PID controller.

Chapter 5

Adaptive MIMO GPC - Theory and Some Implementation Issues

5.1 Introduction

In this chapter the development of a methodology for controlling the flexible manipulator is described. Since the plant parameters are unknown and/or changing, a procedure to identify them is also presented. A good identification method is an integral part of any effective adaptive control algorithm. Without it the controller peformance would be degraded.

The previous chapters showed that actual implementation of control laws on the flexible manipulator has to take into account many unknowns in the manipulator model development. The more important ones are: motor dynamics, moments of inertia of certain components, friction, and signal noise. The model, if available, must also include delays due to algorithm calculations, filters, and motor dynamics. Unfortunately, not enough is known about the model of the flexible manipulator. To make the control problem more difficult, we are faced with a situation where the plant has a smaller number of inputs than outputs. In other words, we have a nonsquare multi-input multi-output (MIMO) system.

The choice of a particular adaptive algorithm depends on expected plant behavior, i.e. whether the open-loop plant is unstable or exhibits non-minimum phase behavior. Additionally, the robustness of the algorithm with respect to plant/model mismatch plays a vital role. The generalized predictive control (GPC) algorithm introduced by Clarke et al. [25] can handle all the above mentioned difficulties and is a very promising choice. It introduces *tuning knobs* (perhaps too many if tuned

independently) by successful integration of ideas from other control algorithms.

Before we develop the MIMO GPC, the single-input single-output (SISO) GPC will be described in some detail so that the MIMO GPC can refer to it whenever necessary. This also enables comparison of the MIMO GPC development and implementation issues with Clarke's SISO GPC.

This chapter is organized as follows. Section 5.2 summarizes the development of the SISO GPC. The extension to MIMO GPC is introduced in section 5.3. Section 5.4 presents algorithms for development of free and forced responses which are parts of the MIMO GPC algorithm. Filtering considerations for the free response of coupled MIMO systems are described in section 5.5. Efficient algorithms for calculating certain components of the MIMO GPC algorithm are explained in section 5.6. The MIMO GPC algorithm involves matrix inversion calculations which are presented in section 5.7. Control constraint considerations for the control of the flexible manipulator using the MIMO GPC are discussed in section 5.8. Effects of output error weighting matrices are explained in section 5.9. The need for consistent control requirements for systems which have fewer inputs than outputs is discussed in section 5.10. Noise filter implementation is introduced in section 5.11. Finally, the MIMO identification procedure to be used in conjunction with the MIMO GPC control algorithm and its implementation are explained in section 5.12.

5.2 SISO GPC

The SISO GPC algorithm is described in great detail by Clarke et al. [25] and Clarke and Mohtadi [24]. Only the basic features are described here.

Although majority of controlled processes have nonlinear dynamics, it is quite usual to describe their behavior around a particular operating point by the linear ARMAX (Autoregressive Moving Averages Exogenous) model

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})\xi(t) \tag{5.1}$$

where t is the discrete time variable denoting the current time instant, $y(t)$ is the sampled output variable, $u(t)$ is the sampled input variable, $\xi(t)$ is uncorrelated random sequence, q^{-1} is the backward shift operator. For simplification purposes, the system delay is incorporated by leading zeros in the $B(q^{-1})$ polynomial. The polynomials

$A(q^{-1})$, $B(q^{-1})$ and $C(q^{-1})$ have the following form

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_A} q^{-n_A} \tag{5.2}$$

$$B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_B} q^{-n_B} \tag{5.3}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \cdots + c_{n_C} q^{-n_C} \tag{5.4}$$

where n_A, n_B, and n_C are the orders of the polynomials $A(q^{-1})$, $B(q^{-1})$, and $C(q^{-1})$.

To accommodate step disturbances, GPC uses ARIMAX (AutoRegressive Integrated Moving Average with an eXogenous input) model

$$A(q^{-1})y(t) = B(q^{-1})u(t) + \frac{T(q^{-1})}{\Delta(q^{-1})}\xi(t) \tag{5.5}$$

where the polynomials $T(q^{-1})$ and $\Delta(q^{-1})$ have the following forms

$$T(q^{-1}) = 1 + t_1 q^{-1} + t_2 q^{-2} + \cdots + t_{n_T} q^{-n_T} \tag{5.6}$$

$$\Delta(q^{-1}) = 1 - q^{-1} \tag{5.7}$$

and n_T is the order of the $T(q^{-1})$ polynomial.

Such a model choice has several implications. First, the noise model is not stationary and represents noise in the form of random step disturbances. This has the advantage that an algorithm based on such a model will guarantee zero steady-state error implying built-in integral action of the algorithm. Second, the ARIMAX model as a member of equation error models allows for model parameterization in the form of pseudolinear regression which is convenient for on-line recursive identification methods. Third, it is a linear model implying that the plant dynamics are linearized around its current operating point. Fourth, although the $T(q^{-1})$ polynomial could be identified on-line, it is not recommended and instead the $T(q^{-1})$ is taken as a tuning knob incorporating designer's knowledge about the noise character. The $T(q^{-1})$ polynomial has a significant effect on plant/model mismatch and its choice can have direct influence on stability of the closed-loop system.

Since GPC is a predictive algorithm, it requires knowledge of j-step ahead output prediction assuming full knowledge of present and past outputs, and future, present, and past control signals. Two approaches to the prediction calculations can be taken. The first one is based on the solution of the Diophantine equation

$$\frac{T(q^{-1})}{A(q^{-1})\Delta(q^{-1})} = E_j(q^{-1}) + q^{-j}\frac{F_j(q^{-1})}{A(q^{-1})\Delta(q^{-1})} \tag{5.8}$$

where $deg(E_j(q^{-1})) = j-1$ and $deg(F_j(q^{-1})) \leq n_A$. If we multiply the ARIMAX model by $q^{-1}E_j(q^{-1})\Delta(q^{-1})$ and use the above Diophantine equation, after some algebraic manipulations we obtain the following equation for j-step ahead prediction

$$y(t+j) = F_j(q^{-1})\frac{y(t)}{T(q^{-1})} + E_j(q^{-1})B(q^{-1})\frac{\Delta(q^{-1})u(t+j)}{T(q^{-1})} + E_j(q^{-1})\xi(t+j) \quad (5.9)$$

Since $\xi(t+j)$ is an uncorrelated random sequence about which nothing can be said at time t for $j > 0$, the optimal j-step ahead prediction of $y(t+j)$ conditioned on knowledge at time t and knowledge of future control signal is

$$\hat{y}(t+j|t) = F_j(q^{-1})\frac{y(t)}{T(q^{-1})} + E_j(q^{-1})B(q^{-1})\frac{\Delta(q^{-1})u(t+j)}{T(q^{-1})} \quad (5.10)$$

Calculation of the output predictions requires the solution of the Diophantine equation 5.8 for $j = 1, \ldots, N_2$ which is quite computationally expensive. Clarke et al. [25] present an efficient recursive procedure for the solution of the j Diophantine equations, but even this procedure is considered relatively computationally expensive. A more efficient method for the prediction calculations is based on successive substitution [59]. Consider the ARIMAX model written as

$$\frac{\Delta(q^{-1})y(t+j)}{T(q^{-1})} = \left\{1 - A(q^{-1})\right\}\frac{\Delta(q^{-1})y(t+j)}{T(q^{-1})} + B(q^{-1})\frac{\Delta(q^{-1})u(t+j)}{T(q^{-1})} + \xi(t+j) \quad (5.11)$$

Note that to calculate the filtered prediction at time $t+j$, we need to know filtered inputs and outputs up to $(t+j-1)$ except for the random variable $\xi(t+j)$ about which nothing can be said at t. Also, the outputs $\Delta(q^{-1})y(t+j-i)/T(q^{-1})$ used in the above equation are either obtained from previous predictions or from available output measurements. Therefore, the equation for filtered predictions is

$$\frac{\Delta(q^{-1})\hat{y}(t+j|t)}{T(q^{-1})} = \left\{1 - A(q^{-1})\right\}\frac{\Delta(q^{-1})\hat{y}(t+j)}{T(q^{-1})} + B(q^{-1})\frac{\Delta(q^{-1})u(t+j)}{T(q^{-1})} \quad (5.12)$$

where $\hat{y}(t+j|t) = \hat{y}(t+j|t)$ if $j > 0$ and $\hat{y}(t+j|t) = y(t+j)$ for $j \leq 0$. Let us define the filtered signals by a subscript F as follows

$$y_F(t+j) = \frac{\Delta(q^{-1})y(t+j)}{T(q^{-1})} \quad \text{and} \quad u_F(t+j) = \frac{\Delta(q^{-1})u(t+j)}{T(q^{-1})} \quad (5.13)$$

The computational procedure is commenced by first calculating the necessary filtered inputs and outputs, and then by j successive substitutions one can obtain all j filtered

predictions. If necessary, these can be defiltered by

$$\hat{y}(t+j|t) = \frac{T(q^{-1})\widehat{y_F}(t+j|t)}{\Delta(q^{-1})} \tag{5.14}$$

Proof that the system of Diophantine equations and the successive use of the equation 5.12 with defiltering yield identical results is presented by Lambert [59]. Since the objective in this thesis is to develop a multi-input multi-output (MIMO) version of GPC which will be considerably more computationally demanding than its SISO counterpart, the method of successive substitutions is used in the remaining part of this thesis.

In the GPC algorithm, the predictions of the controlled system consist of two parts: i) the free response and ii) the forced response. Both parts make sense when the input signal starting from time t is seen as a sum of two control signals. The first one is constant over the entire prediction horizon and equal to the control signal $u(t-1)$. The second part of the control signal is an incremental control sequence describing the control signal variation starting from time t

$$\begin{aligned}
\Delta(q^{-1})u(t) &= u(t) - u(t-1) \\
\Delta(q^{-1})u(t+1) &= u(t+1) - u(t) \\
&= u(t+1) - u(t-1) - \Delta(q^{-1})u(t) \\
\Delta(q^{-1})u(t+2) &= u(t+2) - u(t+1) \\
&= u(t+2) - u(t-1) - \Delta(q^{-1})u(t) - \Delta(q^{-1})u(t+1) \\
\Delta(q^{-1})u(t+j) &= u(t+j) - u(t+j-1) \\
&= u(t+j) - u(t-1) - \textstyle\sum_0^{j-1} \Delta(q^{-1})u(t+j)
\end{aligned} \tag{5.15}$$

The prediction sequence which evolves because of the constant control signal $u(t+j) = u(t-1)$ and the state in which the controlled plant was at the time t and which is defined by $y(t)$, $y(t-1)$, ..., and $u(t-1)$, $u(t-2)$, ..., is the *free response* part $p(t+j)$ of the prediction. *The forced (or driven) response* part $\hat{y}_d(t+j|t)$ of the prediction sequence is defined as the deterministic output of the controlled plant starting with zero initial conditions at time t and driven by the control increment sequence $\Delta(q^{-1})u(t+j)$ for $j \geq 0$.

$$\hat{y}_d(t+j|t) = \frac{B(q^{-1})\Delta(q^{-1})u(t+j)}{A(q^{-1})\Delta(q^{-1})} = \frac{B(q^{-1})}{A(q^{-1})\Delta(q^{-1})}\Delta(q^{-1})u(t+j) \tag{5.16}$$

where the term $B(q^{-1})/(Aq^{-1}\Delta(q^{-1}))$ is the step response of the plant. The above

equation can be written as

$$\hat{y}_d(t+j|t) = \sum_{i=1}^{i=j} g_i \Delta(q^{-1}) u(t+j-i) = G_j(q^{-1})\Delta(q^{-1})u(t+j) \tag{5.17}$$

The elements of the step response polynomial $G_j(q^{-1})$ are obtained by successive substitutions in equation 5.16 with $\Delta(q^{-1})u(t+j) = 1$. The polynomials $G_j(q^{-1})$ for $j = 1, 2, \ldots$ create the step response matrix $\mathbf{G}(q^{-1})$ so that the *key equation* for the output prediction can be written as

$$\mathbf{y} = \hat{\mathbf{y}} + \xi = \mathbf{y_d} + \mathbf{p} + \xi = \mathbf{G}\boldsymbol{\Delta}\mathbf{u} + \mathbf{p} + \xi \tag{5.18}$$

where

$$\xi^T = [\xi(t+1), \xi(t+2), \ldots, \xi(t+j)] \tag{5.19}$$

$$\mathbf{y}^T = [y(t+1), y(t+2), \ldots, y(t+j)] \tag{5.20}$$

$$\mathbf{p}^T = [\hat{p}(t+1|t), \hat{p}(t+2|t), \ldots, \hat{p}(t+j|t)] \tag{5.21}$$

$$\mathbf{y}_d^T = [\hat{y}_d(t+1|t), \hat{y}_d(t+2|t), \ldots, \hat{y}_d(t+j|t)] \tag{5.22}$$

$$\Delta\mathbf{u}^T = [\Delta u(t), \Delta u(t+1), \ldots, \Delta u(t+j-1)] \tag{5.23}$$

where the superscript T means vector transpose. The matrix $\mathbf{G}$ is of the following form

$$\mathbf{G} = \begin{pmatrix} g_1 & & & \\ g_2 & g_1 & & \\ \vdots & \vdots & \ddots & \\ g_j & g_{j-1} & \cdots & g_1 \end{pmatrix} \tag{5.24}$$

Additional tuning knobs are introduced in the form of finite output and input horizons and control weighting in a cost function which has the following form.

$$J = \sum_{j=N_1}^{j=N_2} e^2(t+j) + \lambda_u \sum_{j=1}^{j=NU} \Delta u^2(t+j-1) \tag{5.25}$$

where $N1$ and $N2$ are the initial and final values of the prediction horizons, NU is the control horizon after which the vector of control increments is zero, and λ_u is the control weighting coefficient.

If the prediction error vector $\mathbf{e} = \mathbf{w} - \hat{\mathbf{y}}$ is introduced, where $\mathbf{w}$ is the vector

of future setpoints

$$\mathbf{w}^T = [w(t+1), w(t+2), \ldots, w(t+j)] \tag{5.26}$$

then the cost function can be written in the vector-matrix form as

$$J = (\mathbf{w} - \mathbf{G}\Delta\mathbf{u} - \mathbf{p})^T(\mathbf{w} - \mathbf{G}\Delta\mathbf{u} - \mathbf{p}) + \lambda_u \Delta\mathbf{u}^T\Delta\mathbf{u} \tag{5.27}$$

The optimal control increment sequence is obtained by minimizing J with respect to $\Delta\mathbf{u}$ and is given by

$$\Delta\mathbf{u}^* = (\mathbf{G}^T\mathbf{G} + \lambda_u\mathbf{I})^{-1}\mathbf{G}^T(\mathbf{w} - \mathbf{p}) . \tag{5.28}$$

Although this sequence contains increments over the entire control horizon, the receding horizon philosophy utilizes only the first component of $\Delta\mathbf{u}^*$ and the control calculations are repeated in the next sampling period. This means that the GPC algorithm is *suboptimal.*

The assumption that $\Delta(q^{-1})u(t+j) = 0$ for $j > NU$ defines the dimensions of the matrix $\mathbf{G}$ which are $(N2 - N1 + 1) \times (NU)$. For $NU = 1$, the matrix inversion part of the control calculations is reduced to a division by a scalar.

Choice of different horizons is governed by the controlled plant dynamics and the sampling frequency. Generally, $N1$ should be larger than the plant delay (if known) or can be set to 1 as a default. $N2$ should cover the rise time of the controlled plant even for non-minimum phase systems. Choice of $NU = 1$ is suitable for well behaved plants, otherwise it is recommended that NU be equal to the number of unstable or badly damped poles. The control weighting coefficient should be always nonzero to guarantee nonsingularity of the matrix inversion. However, increased value of λ_u corresponds to less active control effort resulting in more damped behavior.

Additional tuning knobs are available in the form of polynomials $P(q^{-1})$, $R(q^{-1})$, and $Q(q^{-1})$ if an auxiliary plant is introduced as follows

$$\psi(t+j) = P(q^{-1})y(t+j) \tag{5.29}$$

$$\overline{w}(t+j) = R(q^{-1})w(t+j) \tag{5.30}$$

$$\Delta\overline{u}(t+j) = Q(q^{-1})\Delta u(t+j) \tag{5.31}$$

The choice of $P(q^{-1})$, $R(q^{-1})$, and $Q(q^{-1})$ should ensure identical steady-state levels of the original as well as the auxiliary outputs. The cost function can be written then as

$$J = \sum_{j=N_1}^{j=N_2} \{\overline{w}(t+j) - \psi(t+j)\}^2 + \lambda_u \sum_{j=1}^{j=NU} \Delta\overline{u}^2(t+j-1) \tag{5.32}$$

The polynomials $P(q^{-1})$, $R(q^{-1})$, and $Q(q^{-1})$ are not frequently used because more tuning knobs are introduced by this approach.

In general, it is difficult, if not impossible to make analytical statements about stability, transient shape, and other parameters of the closed-loop behavior. Some heuristic or simulated responses are used to develop guidelines or conclusions for the choice of the tuning knobs. The effects of some of the tuning knobs are partially overlapping if considered for a particular dynamics feature. For example, slower response can be achieved by either increasing λ_u or increasing $N2$.

The proper choice of the noise polynomial $T(q^{-1})$ in the GPC algorithm seems to be an open research issue. It is possible to come up with a good choice for simulated examples, but it is much more difficult to do so when dealing with a real plant whose nonlinear dynamics strongly influence its linearized dynamics. Yet, the choice of $T(q^{-1})$ is crucial especially because of the everpresent plant/model mismatch.

5.3 MIMO GPC

It would have been convenient if the extension of SISO GPC algorithm to its MIMO counterpart could be achieved by making necessary notational changes. However, this process is much more involved not only notationally but also algorithmically due to couplings which are not encountered in the SISO algorithm. While implementation issues associated with the dimensionality of the controlled plant are usually manageable in the SISO case, this is not so in the MIMO algorithm and software maintenance and additional improvements of the algorithm can be done effectively only with great effort. This is especially true when unhealthy software engineering practices (shortcuts) are used to minimize the execution time. Additionally, the large number of combinations of the algorithm parameters available to the designer in the MIMO case lead to heuristic choices to achieve acceptable closed-loop system behavior.

The transportation delay (constant or varying) in the SISO systems is frequently a reason why the closed-loop performance deteriorates for many control algorithms. The situation is even more pronounced in the MIMO case where delays exist between each input and each output. In the case of many MIMO algorithms, an interactor matrix is needed. This matrix is based on the knowledge of all the delays in the system. Knowledge of the interactor matrix is equivalent to an almost complete knowledge of the MIMO system dynamics [72]. Shah et al. [88] have proved that for MIMO GPC the knowledge of the interactor matrix is not necessary, as long as $N2 - NU \geq d - 1$. $N2$ is the common value of the final prediction hori-

zon for all outputs, NU is the common value of the control horizon for all inputs, and d is the maximum delay in the system. For the case when $N2_i$'s and NU_j's are different, a stronger and more conservative requirement is obtained in the form $N2_{i_{MIN}} - NU_{j_{MAX}} \geq d - 1$. In most practical situations this condition is satisfied.

The development of the MIMO GPC algorithm is based on MIMO ARIMAX model with n outputs and m inputs which can be described as a set of n MISO (multi-input single-output) subsystems with the i-th subsystem having the following form

$$\sum_{j=1}^{n} A_{ij}(q^{-1})y_j(t) = \sum_{j=1}^{m} B_{ij}(q^{-1})u_j(t) + \frac{T_i(q^{-1})}{\Delta(q^{-1})}\xi_i(t) \tag{5.33}$$

Note that all noise contributions have been lumped into a single term $T_i(q^{-1})\xi_i(t)/\Delta(q^{-1})$ instead of

$$\sum_{j=1}^{n} \frac{T_{ij}(q^{-1})}{\Delta(q^{-1})}\xi_j(t) \tag{5.34}$$

Since $T_{ij}(q^{-1})$ polynomials are not estimated, they will have to be considered as additional tuning knobs. This would result in too many tuning knobs for which no clear choice guideline can be provided.

The above model differs from the MIMO ARIMAX model considered by Tham et al. [98] and Montague et al. [73]. They use a model with P-canonical structure for MIMO systems, which is characterized by $A_{ii}(q^{-1}) \neq 0$. The choice of the P-canonical form is fully justified for square MIMO systems. The P-canonical representation is convenient because it allows for direct application of Diophantine equations as in the SISO GPC with feedforward signals. However, the P-canonical structure does not properly describe the output-coupled non-square systems with $n > m$. Since the flexible manipulator is a non-square system with $n > m$, the fully coupled MIMO ARIMAX model is considered in this work.

Justification of excluding some $A_{ij}(q^{-1})$'s or $B_{ij}(q^{-1})$'s is left up to the identification method. This could be otherwise done properly if a good analytical model were available. Since the implemented adaptive MIMO GPC allows the user to change the structure of the MIMO ARIMAX model as well as orders of each $A_{ij}(q^{-1})$ and $B_{ij}(q^{-1})$ polynomials, it is easy to make several different runs and compare them to find out which coupling effects are weak so that they can be excluded from the model.

If the full MIMO ARIMAX model is accepted, then each MISO subsystem can still be seen as a SISO system with several feedforward signals coming from input and output couplings. This view is convenient if the Diophantine equation approach is taken toward the MIMO system output predictions.

A typical approach to the development of the MIMO GPC algorithm uses the following input and output vector arrangements

$$\mathbf{y}(t+i) = \begin{pmatrix} y_1(t+i) \\ y_2(t+i) \\ \vdots \\ y_n(t+i) \end{pmatrix} \qquad \Delta\mathbf{u}(t+i) = \begin{pmatrix} \Delta u_1(t+i) \\ \Delta u_2(t+i) \\ \vdots \\ \Delta u_m(t+i) \end{pmatrix} \tag{5.35}$$

The following cost function is used.

$$J = \sum_{j=N1}^{j=N2} \{\mathbf{w}(t+j) - \hat{\mathbf{y}}(t+j)\}^T \{\mathbf{w}(t+j) - \hat{\mathbf{y}}(t+j)\} + \lambda_u \sum_{j=1}^{NU} \Delta\mathbf{u}^T(t+j-1)\Delta\mathbf{u}(t+j-1) \tag{5.36}$$

Note, however, that some tuning knobs expected from the MIMO GPC algorithm are missing, particularly the independent output and control horizons for each input and output. Therefore, this direction is not further pursued.

Before continuing with more suitable input and output vector organization, additional tuning knobs will be introduced in the form of prediction error weighting matrices which are described in greater detail in section 5.9. Typically, a error weighting matrix is diagonal and its form for the i-th output is

$$\Lambda_{y_i} = \mathrm{diag}\lambda_{y_{i_j}} \tag{5.37}$$

where j denotes the j-th diagonal element.

Now, to introduce independent horizons and weighting matrices, the input, output, and setpoint vectors are redefined as follows

$$\mathbf{y}_i^T(t) = [y_i(t+N1_i), y_i(t+N1_i+1), \ldots, y_i(t+N2_i)] \tag{5.38}$$

$$\mathbf{w}_i^T(t) = [w_i(t+N1_i), w_i(t+N1_i+1), \ldots, w_i(t+N2_i)] \tag{5.39}$$

$$\Delta\overline{\mathbf{u}}_i^T(t) = [\Delta\overline{u}_i(t), \Delta\overline{u}_i(t+1), \ldots, \Delta\overline{u}_i(t+NU_i-1)] \tag{5.40}$$

Now, the cost function can be written as

$$J = \sum_{i=1}^{i=n} \{\mathbf{w}_i(t) - \hat{\mathbf{y}}_i(t)\}^T \Lambda_{\mathbf{y}_i} \{\mathbf{w}_i(t) - \hat{\mathbf{y}}_i(t)\} + \lambda_u \sum_{i=1}^{i=m} \Delta\overline{\mathbf{u}}_i^T(t)\Delta\overline{\mathbf{u}}_i(t) \tag{5.41}$$

The optimal incremental control sequence for the above cost function is denoted by

$$\Delta\overline{\mathbf{u}}^T(t) = [\Delta\overline{\mathbf{u}}_1^T(t), \Delta\overline{\mathbf{u}}_2^T(t), \ldots, \Delta\overline{\mathbf{u}}_m^T(t)] \tag{5.42}$$

where the i-th incremental vector in the above definition is described by equation 5.40. The *key equation* for each MISO row subsystem can be written as

$$\mathbf{y}_i(t) = \hat{\mathbf{y}}_i(t) + \xi_i(t) = \mathbf{y}_{d_i}(t) + \mathbf{p}_i(t) + \xi_i(t) = \overline{\mathbf{G}}_i\Delta\overline{\mathbf{u}}(t) + \mathbf{p}_i(t) + \xi_i(t) \tag{5.43}$$

To examine the structure of the $\overline{\mathbf{G}}_i$ matrices, let us consider the forced response of a 2×2 system with two sets of output and control horizons.

- a) with $NU_1 = NU_2 = 2$, $N1_1 = N1_2 = 1$, and $N2_1 = N2_2 = 2$

$$\begin{pmatrix} y_{d_1}(t+1) \\ y_{d_1}(t+2) \end{pmatrix} = \begin{pmatrix} g_{11,1} & 0 & g_{12,1} & 0 \\ g_{11,2} & g_{11,1} & g_{12,2} & g_{12,1} \end{pmatrix} \begin{pmatrix} \Delta\overline{u}_1(t) \\ \Delta\overline{u}_1(t+1) \\ \Delta\overline{u}_2(t) \\ \Delta\overline{u}_2(t+1) \end{pmatrix} \tag{5.44}$$

$$\begin{pmatrix} y_{d_2}(t+1) \\ y_{d_2}(t+2) \end{pmatrix} = \begin{pmatrix} g_{21,1} & 0 & g_{22,1} & 0 \\ g_{21,2} & g_{21,1} & g_{22,2} & g_{22,1} \end{pmatrix} \begin{pmatrix} \Delta\overline{u}_1(t) \\ \Delta\overline{u}_1(t+1) \\ \Delta\overline{u}_2(t) \\ \Delta\overline{u}_2(t+1) \end{pmatrix} \tag{5.45}$$

- b) with $NU_1 = 1$, $NU_2 = 2$, $N1_1 = 1$, $N1_2 = 3$, $N2_1 = 3$, and $N2_2 = 4$

$$\begin{pmatrix} y_{d_1}(t+1) \\ y_{d_1}(t+2) \end{pmatrix} = \begin{pmatrix} g_{11,1} & g_{12,1} & 0 \\ g_{11,2} & g_{12,2} & g_{12,1} \end{pmatrix} \begin{pmatrix} \Delta\overline{u}_1(t) \\ \Delta\overline{u}_2(t) \\ \Delta\overline{u}_2(t+1) \end{pmatrix} \tag{5.46}$$

$$\begin{pmatrix} y_{d_2}(t+3) \\ y_{d_2}(t+4) \end{pmatrix} = \begin{pmatrix} g_{21,3} & g_{22,3} & g_{22,2} \\ g_{21,4} & g_{22,4} & g_{22,3} \end{pmatrix} \begin{pmatrix} \Delta\overline{u}_1(t) \\ \Delta\overline{u}_2(t) \\ \Delta\overline{u}_2(t+1) \end{pmatrix} \tag{5.47}$$

This simple example shows that the dimension and content of a particular $\mathbf{G}_i$ matrix depend not only on the MIMO ARIMAX description but also on the prediction and control horizons. This makes the implementation issues more involved.

Assuming for the time being that necessary predictions are available, the op-

timal incremental control sequence which minimizes the cost function is given by

$$\Delta\overline{\mathbf{u}}^*(t) = \left\{\sum_{i=1}^{i=n} \overline{\mathbf{G}}_i^T \mathbf{\Lambda}_{y_i} \overline{\mathbf{G}}_i + \lambda_u \mathbf{I}\right\}^{-1} \left\{\sum_{i=1}^{i=n} \overline{\mathbf{G}}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \tag{5.48}$$

The above result has one shortcoming. When we look at the structure of $\Delta\overline{\mathbf{u}}^*(t)$ (equation 5.42), we can observe that although due to the receding horizon strategy we need only the first component $\Delta\overline{u}_i^*(t)$ of each control vector $\Delta\overline{\mathbf{u}}_i^*(t)$ described by equation 5.40, it is necessary to calculate the entire $\Delta\overline{\mathbf{u}}^*(t)$ to obtain the needed first components. A procedure to overcome this shortcoming is as follows.

Let

$$NU = NU_1 = NU_2 = \ldots = NU_m \tag{5.49}$$

For this case the control increment vector can be rearranged as

$$\Delta\tilde{\mathbf{u}}^T(t) = [\Delta\tilde{u}_1(t), \Delta\tilde{u}_2(t), \ldots, \Delta\tilde{u}_m(t), \Delta\tilde{u}_1(t+1), \ldots, \Delta\tilde{u}_m(t+NU-1),] \tag{5.50}$$

The advantage is that only the first m components of the minimizing incremental control sequence need be calculated, which results in a shorter calculation time. However, the design tuning knob in the form of the variable control horizons is lost. Also, such reorganization of the control vector will result in a different form of the step response matrices which are now represented by $\tilde{\mathbf{G}}_i$'s.

Let us repeat the previous example to see the structure of the $\tilde{\mathbf{G}}_i$ matrices

- $NU_1 = NU_2 = 2$, $N1_1 = N1_2 = 1$, and $N2_1 = N2_2 = 2$

$$\begin{pmatrix} y_{d_1}(t+1) \\ y_{d_1}(t+2) \end{pmatrix} = \begin{pmatrix} g_{11,1} & g_{12,1} & 0 & 0 \\ g_{11,2} & g_{12,2} & g_{11,1} & g_{12,1} \end{pmatrix} \begin{pmatrix} \Delta\tilde{u}_1(t) \\ \Delta\tilde{u}_2(t) \\ \Delta\tilde{u}_1(t+1) \\ \Delta\tilde{u}_2(t+1) \end{pmatrix} \tag{5.51}$$

$$\begin{pmatrix} y_{d_2}(t+1) \\ y_{d_2}(t+2) \end{pmatrix} = \begin{pmatrix} g_{21,1} & g_{22,1} & 0 & 0 \\ g_{21,2} & g_{22,2} & g_{21,1} & g_{22,1} \end{pmatrix} \begin{pmatrix} \Delta\tilde{u}_1(t) \\ \Delta\tilde{u}_2(t) \\ \Delta\tilde{u}_1(t+1) \\ \Delta\tilde{u}_2(t+1) \end{pmatrix} \tag{5.52}$$

The corresponding cost function is

$$J = \sum_{i=1}^{i=n} \{\mathbf{w}_i(t) - \tilde{\mathbf{G}}_i \tilde{\mathbf{u}}(t) - \mathbf{p}_i(t)\}^T \mathbf{\Lambda}_{\mathbf{y}_i} \{\mathbf{w}_i(t) - \tilde{\mathbf{G}}_i \tilde{\mathbf{u}}(t) - \mathbf{p}_i(t)\} + \Delta\tilde{\mathbf{u}}^T(t)\Delta\tilde{\mathbf{u}}(t) \quad (5.53)$$

the minimizing incremental control sequence is

$$\Delta\tilde{\mathbf{u}}^*(t) = \left\{\sum_{i=1}^{i=n} \tilde{\mathbf{G}}_i^T \mathbf{\Lambda}_{y_i} \tilde{\mathbf{G}}_i + \lambda_u \mathbf{I}\right\}^{-1} \left\{\sum_{i=1}^{i=n} \tilde{\mathbf{G}}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \quad (5.54)$$

However, if variable control horizons are also desired the following structure of the vector of control increments can be used

$$\Delta\mathbf{u}^T(t) = [\Delta u_1(t), \Delta u_2(t), \ldots, \Delta u_m(t), \mathbf{u_{rem}}] \quad (5.55)$$

where $\mathbf{u_{rem}}$ denotes the all remaining components, and strongly depends on the choice of the control horizons. Then, just by changing notation, we obtain

$$J = \sum_{i=1}^{i=n} \{\mathbf{w}_i(t) - \mathbf{G}_i \mathbf{u}(t) - \mathbf{p}_i(t)\}^T \mathbf{\Lambda}_{\mathbf{y}_i} \{\mathbf{w}_i(t) - \mathbf{G}_i \mathbf{u}(t) - \mathbf{p}_i(t)\} + \Delta\mathbf{u}^T(t)\Delta\mathbf{u}(t) \quad (5.56)$$

$$\Delta\mathbf{u}^*(t) = \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}\right\}^{-1} \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \quad (5.57)$$

Let us look again at the example with this latest arrangement of control vector increments.

- $NU_1 = 1$, $NU_2 = 2$, $N1_1 = 1$, $N1_2 = 3$, $N2_1 = 2$, and $N2_2 = 4$

$$\begin{pmatrix} y_{d_1}(t+1) \\ y_{d_1}(t+2) \end{pmatrix} = \begin{pmatrix} g_{11,1} & g_{12,1} & 0 \\ g_{11,2} & g_{12,2} & g_{12,1} \end{pmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \\ \Delta u_2(t+1) \end{pmatrix} \quad (5.58)$$

$$\begin{pmatrix} y_{d_2}(t+3) \\ y_{d_2}(t+4) \end{pmatrix} = \begin{pmatrix} g_{21,3} & g_{22,3} & g_{22,2} \\ g_{21,4} & g_{22,4} & g_{22,3} \end{pmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \\ \Delta u_2(t+1) \end{pmatrix} \quad (5.59)$$

This latest version of the MIMO GPC provides the desired flexibility in all tuning knobs and minimizes the step response matrices and multiplications with zeros.

There are two common features for all $\mathbf{G}_i$ matrices and they should be properly exploited. First, number of columns is identical for all $\mathbf{G}_i$'s. Second, only the

components in the first m columns of $\mathbf{G}_i$'s must be calculated because the components in the remaining columns already exist in the first m-columns.

The basic difficulty is that the latest arrangement of $\Delta\mathbf{u}(t)$ results in $\mathbf{G}_i$'s which have seemingly irregular structure which depends on the control horizons chosen. Ordering of components in $\Delta\mathbf{u}(t)$ corresponds to reordering of columns of $\tilde{\mathbf{G}}_i$'s and leaving out those columns for which the appropriate components of $\tilde{\mathbf{u}}(t)$ do not exist.

The general conclusion concerning the structure of $\tilde{\mathbf{G}}_i$'s and $\mathbf{G}_i$'s is that only the first m columns are always ordered independently. Since the order of the remaining columns is a function of the chosen control horizons, it is convenient to define two vectors which directly reflect the structure of control horizons. First, let us define a variable that holds a value equal to the sum of all control horizons, i.e. dimension of the incremental control vector

$$NU_{sum} = \sum_{i=1}^{m} NU_i \tag{5.60}$$

The first vector, called $NU_{ordering}$, is $NU_{sum} \times 1$ and its components indicate to which control horizon the corresponding element of $\Delta\mathbf{u}(t)$ belongs. The second vector, called NU_{time}, is also $NU_{sum} \times 1$ and holds information about the relative time instant when the corresponding component of $\Delta\mathbf{u}(t)$ should be applied. These two vectors, therefore, tell us all needed the information about each component of the incremental control vector $\Delta\mathbf{u}(t)$. An example is provided in section 5.6.

Before continuing with other aspects of the development of the MIMO GPC, it is necessary to describe algorithms for calculating the step coefficients and free responses for the MIMO GPC.

5.4 Free and Forced Responses for MIMO GPC

As in the SISO GPC case, the *key equation* for k-th output can be written as

$$\mathbf{y}_k(t) = \hat{\mathbf{y}}_k(t) + \xi_k(t) = \mathbf{y}_{d_k}(t) + \mathbf{p}_k(t) + \xi_k(t) = \mathbf{G}_k\Delta\mathbf{u}(t) + \mathbf{p}_k(t) + \xi_k(t) \tag{5.61}$$

with

$$\begin{array}{rcl} \mathbf{y}_k^T(t) & = & [\ y_k(t+1) \quad y_k(t+2) \quad \ldots \quad y_k(t+j)\ ] \\ \mathbf{y}_{d_k}^T(t) & = & [\ y_{d_k}(t+1|t) \quad y_{d_k}(t+2|t) \quad \ldots \quad y_{d_k}(t+j|t)\ ] \\ \mathbf{p}_k^T(t) & = & [\ p_k(t+1|t) \quad p_k(t+2|t) \quad \ldots \quad p_k(t+j|t)\ ] \\ \xi_k^T(t) & = & [\ \xi k(t+1) \quad \xi k(t+2) \quad \ldots \quad \xi k(t+j)\ ] \end{array} \tag{5.62}$$

and

$$\Delta \mathbf{u}^T(t) = [\Delta u_1(t), \Delta u_2(t), \ldots, \Delta u_m(t), \Delta u_1(t+1), \ldots, \Delta u_m(t+j-1)] \tag{5.63}$$

Note, that the $\Delta \mathbf{u}(t)$ has a form needed for output predictions and this form is modified by the GPC algorithm when the control horizons are introduced.

To determine the algorithm for development of both the free and forced responses, we have to start from the MIMO ARIMAX model which has been described for the k-th row MISO subsystem as follows

$$\sum_{r=1}^{n} A_{kr}(q^{-1}) y_r(t) = \sum_{r=1}^{m} B_{kr}(q^{-1}) u_r(t) + \frac{T_k(q^{-1})}{\Delta(q^{-1})} \xi_k(t) \tag{5.64}$$

where

$$\begin{array}{rcl} A_{kk}(q^{-1}) & = & 1 + a_{kk,1}q^{-1} + a_{kk,2}q^{-2} + \ldots + a_{kk,n_{A_{kk}}} q^{-n_{A_{kk}}} \\ & & \text{for } k = 1, 2, \ldots, n \\ A_{kr}(q^{-1}) & = & a_{kr,1}q^{-1} + a_{kr,2}q^{-2} + \ldots + a_{kr,n_{A_{kr}}} q^{-n_{A_{kr}}} \\ & & \text{for } k, r = 1, 2, \ldots, n \text{ and } k \neq r \\ B_{kr}(q^{-1}) & = & b_{kr,1}q^{-1} + b_{kr,2}q^{-2} + \ldots + b_{kr,n_{B_{kr}}} q^{-n_{B_{kr}}} \\ & & \text{for } k = 1, 2, \ldots, n \text{ and } r = 1, 2, \ldots, m \\ T_k(q^{-1}) & = & 1 + t_{k,1}q^{-1} + t_{k,2}q^{-2} + \ldots + t_{k,n_{T_k}} q^{-n_{T_k}} \\ & & \text{for } k = 1, 2, \ldots, n \\ \Delta(q^{-1}) & = & 1 - q^{-1} \end{array}$$

It is also assumed that $T_k(q^{-1})$ is a strictly stable polynomial.

Rewriting the ARIMAX model for the k-th row subsystem, we get

$$\frac{\Delta(q^{-1}) y_k(t+j)}{T_k(q^{-1})} = -\sum_{r=1}^{n} \sum_{i=1}^{n_{A_{kr}}} a_{kr,i} \frac{\Delta(q^{-1}) y_r(t+j-i)}{T_k(q^{-1})}$$

$$+\sum_{r=1}^{m}\sum_{i=1}^{n_{B_{kr}}} b_{kr,i}\frac{\Delta(q^{-1})u_r(t+j-i)}{T_k(q^{-1})} + \xi_k(t+j) \qquad (5.65)$$

We can obtain $\Delta(q^{-1})y_k(t+j)/T_k(q^{-1})$ given all outputs and inputs up to the time $t+j-1$ and the stochastic variable $\xi_k(t+j)$ at the time $t+j$. Nothing is known about $\xi_k(t+j)$ at the time t, hence the prediction of the k-th row subsystem can be obtained as

$$\begin{aligned}\frac{\Delta(q^{-1})\hat{y}_k(t+j)}{T_k(q^{-1})} &= -\sum_{r=1}^{n}\sum_{i=1}^{n_{A_{kr}}} a_{kr,i}\frac{\Delta(q^{-1})y_r(t+j-i)}{T_k(q^{-1})} \\ &\quad +\sum_{r=1}^{m}\sum_{i=1}^{n_{B_{kr}}} b_{kr,i}\frac{\Delta(q^{-1})u_r(t+j-i)}{T_k(q^{-1})} \qquad (5.66)\end{aligned}$$

Note, however, that if coupling terms exist, it is necessary to obtain the predictions for each output at the time $t+j-1$ before predictions for the time $t+j$ can be calculated. It is clear from the above equation that even when different output horizons are employed, all predictions must be calculated up to the time $t+N2_{max}-1$ and only those outputs whose output horizons are equal to $N2_{max}$ must be additionally calculated at the time $t+N2_{max}$. Therefore, the prediction calculations are computationally expensive when the general MIMO ARIMAX model with coupling terms is considered. It should be noted that this method of successive substitutions gives as a result the filtered output, i.e. $y_{F_{kk}}(t+j) = \Delta(q^{-1})y_k(t+j)/T_k(q^{-1})$, and the resulting sequence of filtered predictions must be eventually defiltered to be used in the key equations.

Free Response.

The definition of the free response from the SISO GPC still applies, i.e. the free response for the MIMO GPC algorithm is the part of the output predictions which evolves assuming that $u(t+j) = u(t-1)$ for $j = 0, 1, 2, \ldots$ based on the knowledge of measured outputs up to the time t. The method of obtaining properly filtered inputs and outputs needed for the prediction calculations is described in section 5.5. The definitions and notations for filtered inputs and outputs used in the k-th row subsystem are

$$y_{F_{rk}}(t+j) = \frac{\Delta(q^{-1})y_r(t+j)}{T_k(q^{-1})} \quad \text{for} \quad r = 1, 2, \ldots, n \qquad (5.67)$$

$$u_{F_{rk}}(t+j) = \frac{\Delta(q^{-1})u_r(t+j)}{T_k(q^{-1})} \quad \text{for} \quad r = 1, 2, \ldots, m \qquad (5.68)$$

If we assume that the filtered signals are available then the prediction equation for the k-th row subsystem is

$$y_{F_{kk}}(t+j) = -\sum_{r=1}^{n}\sum_{i=1}^{n_{A_{kr}}} a_{kr,i}y_{F_{rk}}(t+j-i) + \sum_{r=1}^{m}\sum_{i=1}^{n_{B_{kr}}} b_{kr,i}u_{F_{rk}}(t+j-i) \qquad (5.69)$$

Now, the entire algorithm can be written as

$$\begin{array}{lll} \textbf{for} & j = 1:1:N2_{max} & \\ \quad \textbf{for} & k = 1:1:n & \\ & y_{F_{kk}}(t+j) & = -\sum_{r=1}^{n}\sum_{i=1}^{n_{A_{kr}}} a_{kr,i}y_{F_{rk}}(t+j-i) \\ & & \quad +\sum_{r=1}^{m}\sum_{i=1}^{n_{B_{kr}}} b_{kr,i}u_{F_{rk}}(t+j-i) \\ \quad \textbf{end} & & \\ \textbf{end} & & \end{array} \qquad (5.70)$$

However, this symbolic notation only describes the algorithm but the implementation issues, i.e. data structures, efficient implementation etc., are omitted. To obtain the defiltered free response which is needed for the *key equation*, the defiltering must be performed as follows

$$\hat{p}_k(t+j) = \frac{T_k(q^{-1})}{\Delta(q^{-1})} y_{F_{kk}}(t+j) \quad \text{for} \quad j = 1,2,\ldots,N2_k \qquad (5.71)$$

Forced Response.

The predicted sequence of the forced response which evolves from zero initial conditions at the time t and is driven by the control increment vector $\mathbf{\Delta u}(t+j)$, where $j \geq 0$, can be again derived by successive substitution in equation 5.66. Clarke and Mohtadi [24] show from the Diophantine equation that the first j-terms of the forced response is equal to the step response of the system driven by the incremental control vector and, importantly, therefore does not depend on the noise polynomial $T(q^{-1})$. The same conclusion of forced response being independent of $T(q^{-1})$ can be obtained according to the reasoning given by Ljung [61] which says that the $T(q^{-1})$ is used to obtain the best prediction given noisy measurements. This means that although the relationship between input and output history does not exactly match the known deterministic model, having some knowledge about the noise character, inputs, and outputs, we can obtain the "best" prediction. When the definition of the forced response is considered, we can see that no noise comes into the forced response prediction calculations because the initial conditions are strictly zero and the prediction calculations evolve exactly following the deterministic part of the MIMO ARIMAX

model.

Because the forced response is driven by the control increment vector, we need, as in the SISO GPC, to develop an algorithm to calculate step response coefficients. The MIMO input and output coupled ARX model is more involved and it is necessary to iterate this model with the input being 1 to get the complete set of MIMO ARX step coefficients. The step response coefficients will be first obtained for the following 2×2 model. The results will then be extended to a general case. The model

$$\begin{pmatrix} A_{11}(q^{-1}) & A_{12}(q^{-1}) \\ A_{21}(q^{-1}) & A_{22}(q^{-1}) \end{pmatrix} \begin{pmatrix} y_1(t+j) \\ y_2(t+j) \end{pmatrix} = \begin{pmatrix} B_{11}(q^{-1}) & B_{12}(q^{-1}) \\ B_{21}(q^{-1}) & B_{22}(q^{-1}) \end{pmatrix} \begin{pmatrix} u_1(t+j) \\ u_2(t+j) \end{pmatrix} \tag{5.72}$$

can be rewritten into the iterative form as

$$\begin{array}{rcllll} y_1(t+j) & = & (1-A_{11})y_1(t+j) & -A_{12}y_2(t+j) & +B_{11}u_1(t+j) & +B_{12}u_2(t+j) \\ y_2(t+j) & = & -A_{21}y_1(t+j) & +(1-A_{22})y_2(t+j) & +B_{21}u_1(t+j) & +B_{22}u_2(t+j) \end{array} \tag{5.73}$$

To develop step coefficients due to $\Delta u_1(t)$, we will set $u_1(t+j) = 1$ for $j = 1, 2, \ldots, N2_{max} - 1$ and $u_2(t+j) = 0$ for $j \geq 0$. The initial conditions are $y_1(t) = y_2(t) = 0$. By repeated application of equation 5.73 we get

t+1:

$$\begin{array}{rcl} y_1(t+1) & = & b_{11}u_1(t) = b_{11,1} = g_{11,1} \\ y_2(t+1) & = & b_{21}u_1(t) = b_{21,1} = g_{21,1} \end{array} \tag{5.74}$$

t+2:

$$\begin{aligned} y_1(t+2) &= -a_{11,1}y_1(t+1) - a_{12,1}y_2(t+1) + b_{11,1}u_1(t+1) + b_{11,2}u_1(t) \\ &= -a_{11,1}g11,1 - a_{12,1}g_{21,1} + b_{11,1} + b_{11,2} \\ &= g_{11,2} \end{aligned}$$

$$\begin{aligned} y_2(t+2) &= -a_{21,1}y_1(t+1) - a_{22,1}y_2(t+1) + b_{21,1}u_1(t+1) + b_{21,2}u_1(t) \\ &= -a_{21,1}g11,1 - a_{22,1}g_{21,1} + b_{21,1} + b_{21,2} \\ &= g_{21,2} \end{aligned}$$

t+3:

$$\begin{aligned} y_1(t+3) &= -a_{11,1}y_1(t+2) - a_{11,2}y_1(t+1) - a_{12,1}y_2(t+2) - a_{12,1}y_2(t+1) \\ &\quad + b_{11,1} + b_{11,2} + b_{11,3} \end{aligned}$$

$$= -a_{11,1}g_{11,2} - a_{11,2}g_{11,1} - a_{12,1}g_{21,2} - a_{12,2}g_{21,1} + b_{11,1} + b_{11,2} + b_{11,3}$$
$$= g_{11,3}$$

$$\begin{aligned} y_2(t+3) &= -a_{21,1}y_1(t+2) - a_{21,2}y_1(t+1) - a_{22,1}y_2(t+2) - a_{22,2}y_2(t+1) \\ &\quad + b_{21,1} + b_{21,2} + b_{21,3} \\ &= -a_{21,1}g_{11,2} - a_{21,2}g_{11,1} - a_{22,1}g_{21,2} - a_{22,2}g_{21,1} + b_{21,1} + b_{21,2} + b_{21,3} \\ &= g_{21,3} \end{aligned}$$

t+4:

$$\begin{aligned} y_1(t+4) &= -a_{11,1}y_1(t+3) - a_{11,2}y_1(t+2) - a_{11,3}y_1(t+1) \\ &\quad - a_{12,1}y_2(t+3) - a_{12,2}y_2(t+2) - a_{12,3}y_2(t+1) \\ &\quad + b_{11,1} + b_{11,2} + b_{11,3} + b_{11,4} \\ &= -a_{11,1}g_{11,3} - a_{11,2}g_{11,2} - a_{11,3}g_{11,1} - a_{12,1}g_{21,3} - a_{12,2}g_{12,2} - a_{12,3}g_{12,1} \\ &\quad + b_{11,1} + b_{11,2} + b_{11,3} + b_{11,4} \\ &= g_{11,4} \end{aligned}$$

$$\begin{aligned} y_2(t+4) &= -a_{21,1}y_1(t+3) - a_{21,2}y_1(t+2) - a_{21,3}y_1(t+1) \\ &\quad - a_{22,1}y_2(t+3) - a_{22,2}y_2(t+2) - a_{22,3}y_2(t+1) \\ &\quad + b_{21,1} + b_{21,2} + b_{21,3} + b_{21,4} \\ &= -a_{21,1}g_{11,3} - a_{21,2}g_{11,2} - a_{21,3}g_{11,1} - a_{22,1}g_{21,3} - a_{22,2}g_{21,2} - a_{22,3}g_{21,1} \\ &\quad + b_{21,1} + b_{21,2} + b_{21,3} + b_{21,4} \\ &= g_{21,4} \end{aligned}$$

From this development we can generalize the $g_{11,j}$ and $g_{21,j}$ step coefficients as

$$g_{11,j} = y_1(t+j) = -\sum_{i=1}^{j-1} a_{11,i}g_{11,j-i} - \sum_{i=1}^{j-1} a_{12,i}g_{21,j-i} + \sum_{i=1}^{j} b_{11,i} \qquad (5.75)$$

$$g_{21,j} = y_2(t+j) = -\sum_{i=1}^{j-1} a_{21,i}g_{11,j-i} - \sum_{i=1}^{j-1} a_{22,i}g_{21,j-i} + \sum_{i=1}^{j} b_{21,i} \qquad (5.76)$$

The general algorithm for the development of step response coefficients due to

all existing inputs is given by

for $r = 1:1:m$ % for all inputs
  for $k = 1:1:n$ % initialize
    $g_{kr,1} = b_{kr,1}$
  end
  for $j = 2:1:N2_{max}$ % for max horizon
    for $k = 1:1:n$ % for all outputs
      $g_{kr,j} = -\sum_{s=1}^{n}\sum_{i=1}^{j-1} a_{ks,i}g_{sr,j-i} + \sum_{i=1}^{j} b_{kr,i}$
    end
  end
end

The step coefficients obtained from the above algorithm can be arranged into a polynomial matrix so that the predictions of the forced output at the time instant $(t+j)$ are

$$\begin{pmatrix} y_{d_1}(t+j) \\ y_{d_2}(t+j) \\ \vdots \\ y_{d_n}(t+j) \end{pmatrix} = \begin{pmatrix} G_{11}(q^{-1}) & G_{12}(q^{-1}) & \dots & G_{1m}(q^{-1}) \\ G_{21}(q^{-1}) & G_{22}(q^{-1}) & \dots & G_{2m}(q^{-1}) \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1}(q^{-1}) & G_{n2}(q^{-1}) & \dots & G_{nm}(q^{-1}) \end{pmatrix} \begin{pmatrix} \Delta u_1(t+j) \\ \Delta u_2(t+j) \\ \vdots \\ \Delta u_m(t+j) \end{pmatrix} \tag{5.77}$$

Each $G_{ij}(q^{-1})$ has the following form

$$G_{ij}(q^{-1}) = g_{ij,1}q^{-1} + g_{ij,2}q^{-2} + \ldots + g_{ij,N2_{max}}q^{-N2_{max}} \tag{5.78}$$

Although the notation is compact, it is not suitable for real-time applications. The conclusions about the actual needs can be obtained by looking at the output of $y_{d_1}(t+j)$ for $j = 1,2,3$ with 3 inputs ($m = 3$) having identical control horizon $NU = 2$. The forced response is calculated from the following equation

$$\begin{pmatrix} y_{d_1}(t+1) \\ y_{d_1}(t+2) \\ y_{d_1}(t+3) \end{pmatrix} = \begin{pmatrix} g_{11,1} & g_{12,1} & g_{13,1} & 0 & 0 & 0 \\ g_{11,2} & g_{12,2} & g_{13,2} & g_{11,1} & g_{12,1} & g_{13,1} \\ g_{11,3} & g_{12,3} & g_{13,3} & g_{11,2} & g_{12,2} & g_{13,2} \end{pmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \\ \Delta u_3(t) \\ \Delta u_1(t+1) \\ \Delta u_2(t+1) \\ \Delta u_3(t+1) \end{pmatrix} \tag{5.79}$$

where the first three columns are the appropriate step coefficients from $G_{11}(q^{-1})$, $G_{12}(q^{-1})$, and $G_{13}(q^{-1})$, respectively. The additional columns, if they exist, are always obtained from the first m columns. The coefficients of the polynomial $G_{ij}(q^{-1})$ can be set up as a $1 \times N2_{max}$ vector. If $G_{ij}(q^{-1})$ is used with any time function ($\Delta u(t)$ etc.) the result is a scalar product as shown below.

$$G_{ij}(q^{-1})\Delta u(t+j) = \begin{pmatrix} g_{ij,1} & g_{ij,2} & \cdots & g_{ij,N2_{max}} \end{pmatrix} \begin{pmatrix} \Delta u(t+j-1) \\ \Delta u(t+j-2) \\ \cdots \\ \Delta u(t+j-N2_{max}) \end{pmatrix} \tag{5.80}$$

The previous arrangement of all G_{ij}'s in equation 5.77 is not the most convenient from the point of view of algorithm implementation. Therefore, a matrix $\mathcal{G}(q^{-1})$ is defined which has the following form

$$\mathcal{G}(q^{-1}) = \begin{pmatrix} G_{11}(q^{-1}) \\ G_{12}(q^{-1}) \\ \cdots \\ G_{1m}(q^{-1}) \\ G_{21}(q^{-1}) \\ G_{22}(q^{-1}) \\ \cdots \\ G_{nm}(q^{-1}) \end{pmatrix} = \begin{pmatrix} g_{11,1}q^{-1} & g_{11,2}q^{-2} & \cdots & g_{11,N2_{max}}q^{-N2_{max}} \\ g_{12,1}q^{-1} & g_{12,2}q^{-2} & \cdots & g_{12,N2_{max}}q^{-N2_{max}} \\ \cdots & \cdots & \cdots & \cdots \\ g_{1m,1}q^{-1} & g_{1m,2}q^{-2} & \cdots & g_{1m,N2_{max}}q^{-N2_{max}} \\ g_{21,1}q^{-1} & g_{21,2}q^{-2} & \cdots & g_{21,N2_{max}}q^{-N2_{max}} \\ g_{22,1}q^{-1} & g_{22,2}q^{-2} & \cdots & g_{22,N2_{max}}q^{-N2_{max}} \\ \cdots & \cdots & \cdots & \cdots \\ g_{nm,1}q^{-1} & g_{nm,2}q^{-2} & \cdots & g_{nm,N2_{max}}q^{-N2_{max}} \end{pmatrix} \tag{5.81}$$

To access a particular step coefficient $g_{kr,j}$ in the $\mathcal{G}(q^{-1})$ matrix, we can see the coefficient is located in $\{(k-1)n+r\}$ row and in the j-th column. The $\mathcal{G}(q^{-1})$ matrix is $nm \times N2_{max}$.

5.5 Input and Output Filtering for the Free Response

The equations for output and input filtering

$$y_{F_{rk}} = \frac{\Delta(q^{-1})y_r(t+j)}{T_k(q^{-1})} \quad \text{and} \quad u_{F_{rk}} = \frac{\Delta(q^{-1})u_r(t+j)}{T_k(q^{-1})} \tag{5.82}$$

become in iterative form

$$y_{F_{rk}} = \{1 - T_k(q^{-1})\}y_{F_{rk}} + y_r(t+j) - y_r(t+j-1) \quad (5.83)$$

$$u_{F_{rk}} = \{1 - T_k(q^{-1})\}u_{F_{rk}} + u_r(t+j) - u_r(t+j-1) \quad (5.84)$$

The initial conditions at the time $t = 0$ are treated properly if the unfiltered signals $y_r(t+j)$ and $u_r(t+j)$ are assumed constant for $t+j < 0$. Then, from the assumption that the noise filters are stable with unity gain, the filtered signal can be taken as equal to the unfiltered signals which corresponds to filtering a DC signal through a low-pass filter. Therefore, the vectors holding the filtered signals must be padded with the proper number of their components equal to the order of the corresponding noise polynomial.

Let us assume that we are at time t. Then the following signals are available from current measurements and previous GPC calculations

$$\begin{array}{llll} y_r(t), & y_r(t-1), & \ldots & \ldots \\ & u_r(t-1), & u_r(t-2), & \ldots \\ & y_{F_{rr}}(t-1), & y_{F_{rr}}(t-2), & \ldots \\ & & u_{F_{rr}}(t-2), & u_{F_{rr}}(t-3), \end{array} \quad \begin{array}{l} \ldots \\ \ldots \\ \ldots \\ \ldots \end{array}$$

To calculate $y_{F_{rr}}(t+1)$, the free response algorithm requires additionally that $y_{F_{rr}}(t)$, $u_{F_{rr}}(t)$, and $u_{F_{rr}}(t-1)$ be available.

Using the above recursive filtering equations, it is easy to obtain $y_{F_{rr}}(t)$ because $y_r(t)$ is available. Although a whole set of predicted filtered inputs $u_{F_{rr}}(t+j)$ for $j = -1, 0, 1, 2, \ldots$ is available from previous free response calculations, it must be discarded because it was calculated with the free response assumptions at the time $t-2$, i.e. that $u_r(t+j) = u_r(t-2)$ for $j = -1, 0, 1, \ldots$. However, at the time $t-1$ a new control signal was applied and all filtered control signals $u_{F_{rr}}(t+j)$ for $j = -1, 0, 1, 2, \ldots, (N2_{max} - 1)$ must be recalculated with a similar time-shifted assumption that $u_r(t+j) = u_r(t-1)$ for $j = 0, 1, 2, \ldots, (N2_{max} - 1)$.

To summarize, at time t we have to calculate all $y_{F_{rr}}(t)$ for $r = 1, 2, \ldots, n$ and also all $u_{F_{rr}}(t+j)$ for $j = -1, 0, 1, 2, \ldots, (N2_{max} - 1)$ and $r = 1, 2, \ldots, m$.

The procedure can stop here if all $T_k(q^{-1})$-filters are the same. Additional steps are involved when the $T_k(q^{-1})$-filters are different. Let us consider a 2×2 ARIMAX model

$$A_{11}(q^{-1})\frac{\Delta y_1(t)}{T_1(q^{-1})} + A_{12}(q^{-1})\frac{\Delta y_2(t)}{T_1(q^{-1})} = B_{11}(q^{-1})\frac{\Delta u_1(t)}{T_1(q^{-1})} + B_{12}(q^{-1})\frac{\Delta u_2(t)}{T_1(q^{-1})}$$

$$A_{21}(q^{-1})\frac{\Delta y_1(t)}{T_2(q^{-1})} + A_{22}(q^{-1})\frac{\Delta y_2(t)}{T_2(q^{-1})} = B_{21}(q^{-1})\frac{\Delta u_1(t)}{T_2(q^{-1})} + B_{22}(q^{-1})\frac{\Delta u_2(t)}{T_2(q^{-1})} \tag{5.85}$$

The attention is oriented towards the coupling terms because their signals are filtered by the noise filters which do not belong to their original row subsystem. This means that the free response algorithm must be carefully interleaved with defiltering and filtering after each step increment of the free prediction calculations.

First, we can see that we have to prepare the coupling filtered input signals and keep proper track of them, i.e.

$$u_{F_{21}}(t+j) = \frac{\Delta u_2(t+j)}{T_1(q^{-1})} \quad \text{and} \quad u_{F_{12}}(t+j) = \frac{\Delta u_1(t+j)}{T_2(q^{-1})} \tag{5.86}$$

must be fully developed as in the case of $u_{F_{11}}(t+j)$ and $u_{F_{22}}(t+j)$ which was described before. The situation with filtered coupling output signals is more involved. The free response algorithm for this general case first develops all filtered inputs, i.e. $u_{F_{11}}(t+j)$, $u_{F_{12}}(t+j)$, $u_{F_{21}}(t+j)$, and $u_{F_{22}}(t+j)$ for $j = -1, 0, 1, \ldots, (N2_{max} - 1)$. Again, the filtered coupled inputs must be developed from proper initial conditions and then at the time t we have to update all filtered outputs, i.e. $y_{F_{11}}(t)$, $y_{F_{12}}(t)$, $y_{F_{21}}(t)$, and $y_{F_{22}}(t)$.

At this point we can calculate the one-step ahead filtered free response to obtain $y_{F_{11}}(t+1)$ and $y_{F_{22}}(t+1)$. To start the next iteration of the free response algorithm, we need also $y_{F_{12}}(t+1)$ and $y_{F_{21}}(t+1)$. Therefore, it is necessary to defilter $y_{F_{11}}(t+1)$ and $y_{F_{22}}(t+1)$ so that the defiltered free response is obtained for $j = 1$ from the defiltering equations

$$p_1(t+j) = \frac{T_1(q^{-1})}{\Delta(q^{-1})} y_{F_{11}}(t+j) \quad \text{and} \quad p_2(t+j) = \frac{T_2(q^{-1})}{\Delta(q^{-1})} y_{F_{22}}(t+j) \tag{5.87}$$

which is, of course, calculated as

$$p_1(t+j) = p_1(t+j-1) + T_1(q^{-1}) y_{F_{11}}(t+j) \quad \text{where} \quad p_1(t) = y_1(t) \tag{5.88}$$

$$p_2(t+j) = p_2(t+j-1) + T_2(q^{-1}) y_{F_{22}}(t+j) \qquad p_1(t) = y_1(t) \tag{5.89}$$

Now, we can calculate the filtered coupled predicted outputs needed for the next step

of calculations of the free response

$$y_{F_{12}}(t+1) = \frac{\Delta p_1(t+1)}{T_2(q^{-1})} \quad \text{and} \quad y_{F_{21}}(t+1) = \frac{\Delta p_2(t+1)}{T_1(q^{-1})} \tag{5.90}$$

i.e.

$$y_{F_{12}}(t+1) = \{1 - T_2(q^{-1})\}y_{F_{12}}(t+1) + p_1(t+1) - p_1(t) \tag{5.91}$$

$$y_{F_{21}}(t+1) = \{1 - T_1(q^{-1})\}y_{F_{21}}(t+1) + p_2(t+1) - p_2(t) \tag{5.92}$$

From the above example the following memory allocations must be made

n^2	filtered output vectors
nm	filtered input vectors
n	output vectors
m	input vectors
n	defiltered free response vectors

The free filtered response algorithm is the most computationally expensive section in the MIMO GPC algorithm. The free response is immediately available in the form of the computed defiltered signals $p_i(t+k)$.

5.6 MIMO GPC Algorithm - Some Implementation Issues

In order to determine the factors which must be taken into account to implement the control law described by equation 5.54 which is reproduced again:

$$\Delta \mathbf{u}^*(t) = \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}\right\}^{-1} \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \tag{5.93}$$

let us consider the following more complicated example involving 3×3 MIMO system with following output and control horizons: $N1_1 = 1$, $N2_1 = 4$, $NU_1 = 2$, $N1_2 = 3$, $N2_2 = 5$, $NU_2 = 1$, $N1_3 = 6$, $N2_3 = 7$, and $NU_3 = 3$. The forced responses for each of the 3 outputs are given by

$$\hat{\mathbf{y}}_{d_1}(t) = \mathbf{G}_1 \Delta \mathbf{u}(t) \tag{5.94}$$

$$\hat{\mathbf{y}}_{d_2}(t) = \mathbf{G}_2 \Delta \mathbf{u}(t) \tag{5.95}$$

$$\hat{\mathbf{y}}_{d_3}(t) = \mathbf{G}_3 \Delta \mathbf{u}(t) \tag{5.96}$$

These, in component form, are shown below

$$\begin{pmatrix} y_{d_1}(t+1) \\ y_{d_1}(t+2) \\ y_{d_1}(t+3) \\ y_{d_1}(t+4) \end{pmatrix} = \begin{pmatrix} g_{11,1} & g_{12,1} & g_{13,1} & 0 & 0 & 0 \\ g_{11,2} & g_{12,2} & g_{13,2} & g_{11,1} & g_{13,1} & 0 \\ g_{11,3} & g_{12,3} & g_{13,3} & g_{11,2} & g_{13,2} & g_{13,1} \\ g_{11,4} & g_{12,4} & g_{13,4} & g_{11,3} & g_{13,3} & g_{13,2} \end{pmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \\ \Delta u_3(t) \\ \Delta u_1(t+1) \\ \Delta u_3(t+1) \\ \Delta u_3(t+2) \end{pmatrix} \tag{5.97}$$

$$\begin{pmatrix} y_{d_2}(t+3) \\ y_{d_2}(t+4) \\ y_{d_2}(t+5) \end{pmatrix} = \begin{pmatrix} g_{21,3} & g_{22,3} & g_{23,3} & g_{21,2} & g_{23,2} & g_{23,1} \\ g_{21,4} & g_{22,4} & g_{23,4} & g_{21,3} & g_{23,3} & g_{23,2} \\ g_{21,5} & g_{22,5} & g_{23,5} & g_{21,4} & g_{23,4} & g_{23,3} \end{pmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \\ \Delta u_3(t) \\ \Delta u_1(t+1) \\ \Delta u_3(t+1) \\ \Delta u_3(t+2) \end{pmatrix} \tag{5.98}$$

$$\begin{pmatrix} y_{d_3}(t+6) \\ y_{d_3}(t+7) \end{pmatrix} = \begin{pmatrix} g_{31,6} & g_{32,6} & g_{33,6} & g_{31,5} & g_{33,5} & g_{33,4} \\ g_{31,7} & g_{32,7} & g_{33,7} & g_{31,6} & g_{33,6} & g_{33,5} \end{pmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \\ \Delta u_3(t) \\ \Delta u_1(t+1) \\ \Delta u_3(t+1) \\ \Delta u_3(t+2) \end{pmatrix} \tag{5.99}$$

$$NU_{sum} = 2 + 1 + 3 = 6 \tag{5.100}$$

and the corresponding step coefficient matrix $\mathcal{G}$ for this case has the following form

$$\mathcal{G} = \begin{pmatrix} g_{11,1} & g_{11,2} & g_{11,3} & g_{11,4} & g_{11,5} & g_{11,6} & g_{11,7} \\ g_{12,1} & g_{12,2} & g_{12,3} & g_{12,4} & g_{12,5} & g_{12,6} & g_{12,7} \\ g_{13,1} & g_{13,2} & g_{13,3} & g_{13,4} & g_{13,5} & g_{13,6} & g_{13,7} \\ g_{21,1} & g_{21,2} & g_{21,3} & g_{21,4} & g_{21,5} & g_{21,6} & g_{21,7} \\ g_{22,1} & g_{22,2} & g_{22,3} & g_{22,4} & g_{22,5} & g_{22,6} & g_{22,7} \\ g_{23,1} & g_{23,2} & g_{23,3} & g_{23,4} & g_{23,5} & g_{23,6} & g_{23,7} \\ g_{31,1} & g_{31,2} & g_{31,3} & g_{31,4} & g_{31,5} & g_{31,6} & g_{31,7} \\ g_{32,1} & g_{32,2} & g_{32,3} & g_{32,4} & g_{32,5} & g_{32,6} & g_{32,7} \\ g_{33,1} & g_{33,2} & g_{33,3} & g_{33,4} & g_{33,5} & g_{33,6} & g_{33,7} \end{pmatrix} \tag{5.101}$$

The $\mathcal{G}$ matrix is $nm \times N2_{max}$ which in our case means that it contains 63 elements. Although some elements are not used in the row subsystem step matrices $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$, they are necessary for the forced response algorithm. Note again, that $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$ have the same number of columns.

The $\mathcal{G}$ matrix contains all elements that are needed for creating $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$. The essential question is the development of an efficient algorithm to build the row subsystem step matrices. The approach taken here is based on extracting an arbitrary column of the chosen step response matrix from the step coefficient matrix $\mathcal{G}$. The column extraction is the crucial part of several algorithms that follow and, therefore, it is explained here in detail.

First, the numerical values of NU_{sum}, $NU_{ordering}$, NU_{time} are determined for the example

$$NU_{sum} = 2 + 1 + 3 = 6 \tag{5.102}$$

$$\begin{aligned} NU^T_{ordering} &= [\ 1\ \ 2\ \ 3\ \ 1\ \ 3\ \ 3\ ] \\ NU^T_{time} &= [\ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 2\ ] \end{aligned} \tag{5.103}$$

Let k denote a particular k-th row subsystem, then all step coefficients needed for building of $\mathbf{G}_k$ are found in the $\mathcal{G}$ matrix rows in the interval

$$row \in < (k-1)m + 1, km > \tag{5.104}$$

It is, therefore, convenient to assign the variable row following value $row = (k-1)m$. Next, it is necessary to determine the first and last elements for a particular column of $\mathbf{G}_k$ where $\mathbf{G}_k$ has columns $j \in < 1, NU_{sum} >$. The initial and last elements are

also functions of the prediction horizon for the particular row subsystem

$$i_{init} = N1[k] - NU_{time}[j] \tag{5.105}$$

$$i_{end} = N2[k] - NU_{time}[j] \tag{5.106}$$

where the square brackets are used to denote a particular element of a given vector.

Notice, for example, that if i_{init} is smaller than 1 then the corresponding element $\mathbf{G}_k[i_{init}][j]$ is zero. If this happens, then the variable i_{init} must be initialized to 1. In such a case, the remaining (not assigned) elements in $\mathbf{G}_k$ should be properly initialized to zero. This zero initialization may not be necessary because the matrix $\mathbf{G}_k$ does not exist alone. It is always a part of a product and, therefore, the zero elements only mean that the part of the product does not have to be performed which results in faster execution. The particular row in $\mathcal{G}$ that should be used in conjunction with the previously determined i_{start} and i_{end} variables is

$$j_{row} = row + NU_{ordering}[k] \tag{5.107}$$

Before the retrieval of components from $\mathcal{G}$ can start, it is necessary to check a variable *counter* which says how many variables should be retrieved. If $counter = i_{end} - i_{start} + 1 < 0$ then the particular column in $\mathbf{G}_k$ consists of zeros only.

Since the expression

$$\left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}\right\} \tag{5.108}$$

represents a $NU_{sum} \times NU_{sum}$ symmetric matrix, it sufficient to calculate the upper triangular part by initializing the resulting matrix with $\lambda_u \mathbf{I}$ first. The elements corresponding to $\mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i$ are calculated directly using the algorithm for retrieval of $\mathbf{G}_i$ columns from $\mathcal{G}$ so that there is no need to build $\mathbf{G}_i$'s and the resulting product elements are calculated in a loop and added directly to the resulting matrix. This scheme not only saves some memory but also is faster because no intermediate results are created and no multiplications by zero are performed.

The particular form of $\mathcal{G}$ is well justified now because extracting columns of $\mathbf{G}_i$ is equivalent to retrieval of elements in rows of $\mathcal{G}$ which is much faster in the C programming language than retrieval of columns and is additionally speeded up by access via incrementing pointers which is faster than access via subscripts. Since the retrieval of $\mathbf{G}_i$ columns is needed many times, the form of $\mathcal{G}$ is very crucial. After

the upper triangular part of the matrix is calculated, the above diagonal elements are copied to their symmetric counterparts.

The second expression we have to deal with is

$$\left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \tag{5.109}$$

The final result of this expression is a vector that must be first initialized to zero. Again, there is no need to create or use the $\mathbf{G}_i$ matrices and, instead, the same algorithm for $\mathbf{G}_i$ column retrieval is used because it corresponds to row retrieval of $\mathbf{G}_i^T$. The same benefits which were explained for calculation of $\mathbf{G}_i^T \Lambda_{y_i} \mathbf{G}_i$ apply for the $\mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i - \mathbf{p}_i)$.

The above procedure is designed to take advantage of the C programming language.

5.7 Matrix Inverse and Control Signal Calculations

In this section we discuss methods to numerically compute the control signals according to equation 5.57 (which is repeated here for convenience)

$$\Delta \mathbf{u}(t) = \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}\right\}^{-1} \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \tag{5.110}$$

This involves the calculation of the inverse of the expression in the first set of brackets. The calculation of the expression in the second set of brackets has been discussed in section 5.6.

Two methods are discussed for the calculations of the required inverse: UD factorization method and LU decomposition method.

UD factorization method.

Mohtadi [72] proposes three ways to cope with the inverse of the matrix $(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})$: Gaussian elimination with pivoting, singular value decomposition, and UD factorization method. The Gaussian elimination is considered unsuitable due to its poor numerical properties. The singular value decomposition method is computationally very expensive and not suitable for fast real-time applications such as MIMO GPC. The UD factorization approach is seen as a reasonable compromise between good numerical properties and computational cost.

UD factorization method - SISO case.

The basic equations for covariance matrix update in parameter estimation [63] are

$$\overline{R}(t) = P^{-1}(t) = \overline{R}(t-1) + \alpha_t \psi \psi^T \tag{5.111}$$

$$P(t) = \overline{R}^{-1}(t) = P(t-1) - \frac{P(t-1)\psi\psi^T P(t-1)}{\frac{1}{\alpha_t} + \psi^T P(t-1)\psi} \tag{5.112}$$

Therefore, if $\overline{R}^{-1}(t-1) = P(t-1)$ is known and updates ψ are available, we can obtain $\overline{R}^{-1}(t) = P(t)$.

We partition one part of the GPC law equation as

$$\lambda \mathbf{I} + \mathbf{G}^T\mathbf{G} = \lambda \mathbf{I} + \begin{bmatrix} d_{N_1} & d_{N_1+1} & \dots & d_{N_2} \end{bmatrix} \begin{bmatrix} d_{N_1} \\ d_{N_1+1} \\ \dots \\ d_{N_2} \end{bmatrix} = \lambda \mathbf{I} + \sum_{i=N_1}^{N_2} d_i d_i^T \tag{5.113}$$

where

$$d_i^T = \begin{bmatrix} g_i & g_{i-1} & \dots & g_2 & g_1 & 0 & \dots & 0 \end{bmatrix} \tag{5.114}$$

To apply the UD algorithm for this case, we set

$$P(t-1) = \overline{R}^{-1}(t-1) = \lambda^{-1}\mathbf{I} \tag{5.115}$$

We then call the UD algorithm $(N_2 - N_1 + 1)$ times. The UD algorithm calculates the matrix $(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})^{-1}$ in a factored form $\mathbf{U}\mathbf{D}\mathbf{U}^T$, where $\mathbf{D}$ is a diagonal matrix and $\mathbf{U}$ is an upper triangular matrix. Both $\mathbf{U}$ and $\mathbf{D}$ are stored for efficiency in one matrix. To obtain the desired inverse, it is necessary to extract $\mathbf{U}$ and $\mathbf{D}$ first and then to perform the matrix multiplication $\mathbf{U}\mathbf{D}\mathbf{U}^T$ to get $(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})^{-1}$.

It turns out that it is more convenient to calculate the following expression directly using the UD algorithm

$$\Delta\mathbf{u} = (\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{G}^T(\mathbf{w} - \mathbf{p}) \tag{5.116}$$

as pointed out by Mohtadi [72]. This is because the UD algorithm also returns the Kalman gain (not the normalized Kalman gain)

$$L(t) = P(t-1)\psi(t) \tag{5.117}$$

Therefore, if the UD algorithm is called once more with

$$P(t-1) = \overline{R}^{-1}(t-1) = (\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})^{-1} \quad \text{and} \quad \psi(t) = \mathbf{G}^T(\mathbf{w} - \mathbf{p}) \tag{5.118}$$

we obtain the Kalman gain vector $L(t)$ which is equal to the desired control vector $\Delta\mathbf{u}(t)$, i.e.

$$\Delta\mathbf{u}(t) = L(t) \tag{5.119}$$

UD factorization method - MIMO case.

Let us extend this approach to the MIMO GPC. Consider the equation

$$\Delta\mathbf{u}(t) = \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}\right\}^{-1} \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \tag{5.120}$$

To be able to use Bierman's UD algorithm, we have to first create all $\overline{\mathbf{G}}_k$ matrices as follows

$$\overline{\mathbf{G}}_k = \mathbf{\Lambda}_{y_k}^{\frac{1}{2}} \mathbf{G}_k \quad \Rightarrow \quad \overline{\mathbf{G}}_k^T = \mathbf{G}_k^T \mathbf{\Lambda}_{y_k}^{\frac{1}{2}} \tag{5.121}$$

so that we can write equation 5.120 as

$$\Delta\mathbf{u}(t) = \left\{\sum_{i=1}^{i=n} \overline{\mathbf{G}}_i^T \overline{\mathbf{G}}_i + \lambda_u \mathbf{I}\right\}^{-1} \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t))\right\} \tag{5.122}$$

The expression in the first set of brackets can be factorized, as in the SISO case, as follows

$$\sum_{i=1}^{i=n} \overline{\mathbf{G}}_i^T \overline{\mathbf{G}}_i + \lambda_u \mathbf{I} = \lambda_u \mathbf{I} + \sum_{k=1}^{n} \sum_{i=N1[k]}^{N2[k]} d_{ki} d_{ki}^T \tag{5.123}$$

To calculate the inverse, we have to call the UD factorization routine $\sum_{k=1}^{n}(N2[k] - N1[k] + 1)$ times. The expression in the second set of brackets in equation 5.122 can be calculated as suggested in section 5.6. The last call of UD routine is carried out as in the SISO case to obtain $L(t) = \Delta\mathbf{u}(t)$.

LU decomposition method.

A second method was implemented to determine the matrix inverse. This method uses LU matrix decomposition method, which, in turn, is based on Crout's method with implicit pivoting [81]. The idea is to factorize efficiently any nonsingular square matrix into a product of two matrices **LU** where **L** is a lower triangular matrix with unity diagonal and **U** is an upper triangular matrix.

First, the expression $\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}$ is evaluated as a single matrix using the scheme proposed in section 5.6. Then, using the Crout's method, the LU

factorization is carried out (i.e.).

$$\left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}\right\} = \mathbf{LU} \tag{5.124}$$

To obtain the desired matrix inverse, a sequence of back substitutions is performed to solve the following equations

$$\mathbf{LU}\mathbf{x}_i = \mathbf{e}_i \qquad i = 1, 2, \ldots, NU_{sum} \tag{5.125}$$

where $\mathbf{e}_i$ is the i-th column of the $NU_{sum} \times NU_{sum}$ unity matrix. This means that the inverse is built columnwise and we get

$$\mathbf{X} = \left[\begin{array}{cccc} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_{NU_{sum}} \end{array}\right] = \left\{\sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} \mathbf{G}_i + \lambda_u \mathbf{I}\right\}^{-1} \tag{5.126}$$

The next step is to compute $\Delta u(t)$ using equation 5.110. Let us denote

$$\mathbf{q} = \sum_{i=1}^{i=n} \mathbf{G}_i^T \mathbf{\Lambda}_{y_i} (\mathbf{w}_i(t) - \mathbf{p}_i(t)) \tag{5.127}$$

which can be calculated using the method in section 5.6. The control signal increments are calculated as

$$\Delta\mathbf{u} = \left[\begin{array}{cccc} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_{NU_{sum}} \end{array}\right] \mathbf{q} \tag{5.128}$$

However, equation 5.128 implies that all the NU_{sum} columns have to be computed first and only then the first m rows of $\mathbf{X}$ are used to obtain the desired first m components of the control increment signal. This, however, does not utilize the property that the inverse of a symmetric matrix is again a symmetric matrix. Since, in our case,

$$\left[\begin{array}{cccc} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_{NU_{sum}} \end{array}\right]^T = \left[\begin{array}{c} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \ldots \\ \mathbf{x}_{NU_{sum}}^T \end{array}\right] \tag{5.129}$$

we need to calculate only the first m columns. This will result in saving in computational time.

The UD method is known to possess good numerical properties [15]. Its use is especially recommended if the matrix to be inverted is ill-conditioned. The LU method, on the other hand, is not expected to perform well if the matrix is ill-

conditioned. However, it is faster than the UD method. In case NU_{sum} is small, e.g. ≤ 4, very often a direct algebraic method is more practical and efficient. However, this method becomes unwieldy for larger matrices ($NU_{sum} > 4$). In such cases the LU method is preferred.

5.8 Control Constraints

If no constraints are imposed on the control signal, then an unconstrained control signal results from the GPC algorithm. However, in all practical situations, actuators have amplitude limits due to saturation and rate limits determined by their bandwidth. If these constraints are known, they should be incorporated into the cost function of the control algorithm.

Tsang and Clarke [102] addressed this issue for SISO GPC in two separate cases. First case, for $NU = 2$, with only amplitude constraints and the second case, as well for $NU = 2$, with only rate constraints. Mutha [75] treats rate and amplitude constraints simultaneously for $NU = 2$ in the SISO case and $NU = 1$ in the MIMO case. Large NU's lead to computationally expensive solutions.

For reasons stated below, the above approach was not followed in this project. In the case of the flexible manipulator, at least two badly located poles always exist, requiring $NU = 2$ even if only the first modal frequency is considered. Such a case requires a numerical solution of a constrained quadratic programming problem whose timing requirements are beyond real-time control timing constraints.

Although the direct-drive motor saturation occurs for the control signal voltage equal to ± 10V, any algorithm which would generate this maximum signal for more than 2 sampling periods is not practical. If this happens, the motor would deliver an excessive torque and drive the manipulator outside its workspace which results in activating the emergency brakes. It is therefore necessary to define amplitude limits less than ± 10V. If the control signal tries to sustain the amplitude over this lower limit for more than 3 sampling periods, the experiment should be aborted.

It is not generally desirable to increase the control weight coefficient or to reduce the control horizon NU, because these result in more sluggish response of the controlled system. However, such an alternative is acceptable in the case of the flexible manipulator. Less energetic control effort is not as likely to excite higher modal frequencies of the flexible link, which is very desirable.

5.9 Output Error Weighting Coefficients

In using GPC or for that matter any algorithm which uses a cost function minimization approach to control a system, several factors have to be taken into account. These are discussed below. The effects of some of these will become more important if the system has less inputs than outputs ($m < n$) as is the case with the flexible manipulator. Output error weighting coefficients have an influence on many of these factors. In the case of the MIMO GPC, Λ_{y_i} respresents the output error weighting matrices (equation 5.37).

De Keyser et al. [29] introduced output error weighting coefficients in SISO case on the base of emphasizing different sections of the prediction horizons, while Mutha [75] introduced them to equalize prediction error amplitudes and to influence steady-state errors of different outputs. This work introduced the output error weighting coefficients independently of the quoted references because it was obvious from the onset that the control of the flexible manipulator must deal with the problem of smaller number of inputs than outputs, i.e. $m < n$. Then the weighting matrices help to define which outputs are more important and should closely follow their prespecified trajectories if the nature of the system allows it. Also, confidence in sensor information can be projected into the weighting matrix structure. Overall, the reasons for introducing the diagonal weighting matrices Λ_{y_i} are summarized :

- Steady state deviation of the output signals from the desired values. This is particularly important for nonsquare systems where $m < n$. A conscious decision must be made as to which outputs should be made to follow the desired steady state levels more closely than others.

- The noise content of measurements from the different sensors may not be the same. High noise content will influence the output predictions.

- Since the control effort is determined by the absolute value of the error in the different outputs, control effort will more likely respond to the errors of larger signals. If a system has two outputs, one large and one small, and both have the same signal/noise ratio, the control algorithm will be biased toward the larger signal but in a particular case the smaller signal may be of higher importance.

- The accuracy of the model is important for the output predictions. Inaccurate models result in larger prediction errors which in turn have direct impact on the control effort.

- It is important to monitor and control the transient behavior of the process.

Sound engineering judgement must be exercised with respect to the choice of relative and absolute values of elements of the Λ_{y_i}'s. The relative values are important when the major focus is on system behavior (even in the deterministic case) while the proper choice of absolute values of Λ_{y_i}'s is important for dealing with plant/model mismatch.

Although the output error weighting coefficients are introduced in the form of diagonal matrices in the development of the GPC algorithm they are in reality implemented as vectors to save memory and to speed up the access time. They are implemented as

$$\lambda y_{i_{jj}} = \lambda y_{i_{j-1,j-1}} \kappa_i \tag{5.130}$$

where i defines the particular output and the lowest subscript defines the position in the Λy_i matrix. The variable κ_i defines whether we trust distant predictions more than the closer predictions which is expressed by a choice $\kappa_i \geq 1$. If we do not have the confidence in the distant predictions then $\kappa_i \in (0, 1>$.

The first part of the MIMO GPC cost function (equation 5.57) with output error weighting coefficients has the following form

$$\sum_{i=1}^{n} \sum_{j=N1_i}^{N2_i} \lambda_{y_{i_{jj}}} e^2(t+j) \tag{5.131}$$

which has a certain resemblance to the weighted least square criterion

$$V(N,\theta) = \frac{1}{N} \sum_{t=1}^{N} \alpha_t e^2(t) \tag{5.132}$$

However, with the latter we try to minimize past and present errors while in the case of GPC we try to minimize the future errors - property of the predictive formulation of the GPC cost function. The choice of the output weighting error coefficients is specified by the values of $\lambda_{y_{i_{11}}}$ and κ_i. Different choices for these two parameters for the i-th output will have different impact on the system response.

The actual implementation is described by the following equations

$$\lambda_{y_{i_{jj}}} = \lambda_{y_{i_{11}}} \quad \text{for} \quad j \leq N1_i \tag{5.133}$$

$$\lambda_{y_{i_{jj}}} = \lambda_{y_{i_{j-1,j-1}}} \kappa_i \quad \text{for} \quad N1_i < j \leq N2_i \tag{5.134}$$

Proper choice of $\lambda_{y_{i_{11}}}$'s and κ_i's requires knowledge of the controlled system. Some

of these choices will be demonstrated later via simulations.

5.10 Nonsquare Systems and MIMO GPC

MIMO systems which do not have the same number of inputs and outputs are called nonsquare systems. The situation when more inputs exist, i.e. $m > n$, can be handled by the MIMO GPC as long as a reasonable model of the controlled plant is available. Although it seems that redundant inputs are available, the option of not using the redundant inputs is system dependent because it depends on coupling terms of the model. A situation can be conceived when eliminating "redundant" inputs can result in too large control signals if the performance requirements are to be met. It is, therefore, safer to provide all available inputs to the MIMO GPC law so that an unconstrained solution can be obtained. The resulting control signal amplitudes indicate which inputs are treated as less important by the MIMO GPC.

In real applications, the case of smaller number of inputs than outputs, i.e. $m < n$, is by far more frequent. Since this means a loss of some degrees of freedom, compromises in control peformance must be accepted. Let us consider a system with 2 outputs and one input. If the transient and steady state control requirements are different but not incompatible then at least a compromise performance level can be achieved with a good control law. If, however, the requirements are contradictory then an acceptable compromise may not be possible. Fortunately, in the case of the flexible manipulator the performance specifications are generally given in the form of trajectory following. The flexible tip should follow the specified trajectory while the tip vibrations should be minimized. This is physically feasible because in the extreme case the trajectory could be defined as a slow smooth curve which would minimize the chance of exciting the flexible link modal frequencies.

The use of output error weighting coefficients, discussed in some detail in section 5.9, was also motivated by the need to specify the requirements for trajectory following for systems characterized by $m < n$.

5.11 T Filter

The design of the $T(q^{-1})$ filter is still an active research area [83]. Its choice depends on the plant/model mismatch, nonlinearities, and plant noise and observation noise. $T(q^{-1})$-filters enable us to mitigate the effects of the deviations from ideal conditions. Without them, output predictions would be significantly different from predictions

based on an accurate model and noise-free inputs and outputs. By properly choosing $T(q^{-1})$ we can obtain predictions which describe the dominant behavior.

The $T(q^{-1})$ filter is used in the free response prediction. First, it filters all inputs and outputs to eliminate high frequency glitches (if, as usual, the $T(q^{-1})$ filter has a low-pass filter character) and these filtered signals are then used for the filtered free prediction algorithm. Second, the predicted filtered free response is then defiltered using the filtered free response and the measured outputs up to the time t. This, however, means that measurement noise propagates into the predicted free response via $T(q^{-1})/\Delta(q^{-1})$ and increases the inaccuracy of the predicted free response. The GPC $T(q^{-1})$ filter has no effect on the plant model because it does not modify any $a_{ij,k}$ or $b_{ij,k}$ coefficients. The improvement of the plant model must come from the identification method. The GPC $T(q^{-1})$ filter does not try to correct model deficiencies. This is not to say that the GPC $T(q^{-1})$ filter is not very useful, on the contrary, sometimes a good choice of the $T(q^{-1})$ filter is the only way to make the closed-loop system stable.

The overall aim of the GPC $T(q^{-1})$ filter is to reduce the effects of the plant/model mismatch and noisy inputs and outputs so that errors in prediction calculations are minimized. Interestingly, similar effects, perhaps in more ad hoc fashion, can be obtained via output error weighting coefficients as described earlier. While in the case of the $T(q^{-1})$ filters we are concerned with the frequency domain, the use of the output error weighting coefficients is related to the time domain.

The choice of $1/T(q^{-1})$ as a low-pass filter is justified by the usual assumption that noise effects exhibit high frequency behavior and should be filtered out by an appropriate low-pass filter.

Since the flexible link theoretically exhibits infinite number of modal frequencies and the motor bandwidth is limited, a deliberate choice must be made about frequency range of input and output signals. A typical choice of the $T(q^{-1})$ filter is $1/T(q^{-1}) = 1/(1-\alpha q^{-1})^i$ [23] where i is an integer number. For $i = 1$ the analog prototype has the form $1/(sj\omega + a)$ where a is the corner frequency. The corner frequency for the corresponding digital low-pass filter $1/(1 - e^{-ah}z^{-1})$ with h being the sampling period, is given by $a = -(\ln \alpha)/h$ in rad/sec.

The values of the corner frequencies are shown in Table 5.1 so that the effect of α can be estimated for a particular sampling frequency. The f_f frequency is the folding frequency, i.e. $f_f = 1/(2h)$

h [sec]	f_f [Hz]	$\alpha = 0.99$	$\alpha = 0.9$	$\alpha = 0.5$	$\alpha = 0.1$
1	0.5	0.01	0.105	0.69	0.23
0.1	5	0.1	1.05	6.9	2.3
0.02	25	0.5	5.26	34.6	115.1
0.01	50	1.005	10.53	69.31	230.2

Table 5.1: Corner frequencies [rad/sec].

5.12 Identification Procedure for MIMO GPC

The discussion so far in this chapter assumes that the plant model is known. However in many practical situations this will not be the case. Sometimes, only partial information regarding the plant parameters will be available and at other times no information may be available. It is therefore necessary to have a procedure in place which will identify the unknown plant parameters based on measured input and output data. In the case of the flexible manipulator, it is difficult to arrive analytically at a model which adequately describes the highly nonlinear and coupled dynamics of the manipulator. The fact that the manipulator is a MIMO plant makes such a task more difficult. A procedure for identification of the parameters of the manipulator is developed in this section. This identification procedure will be used in conjunction with the MIMO GPC algorithm.

The majority of available identification procedures are presented for SISO systems. While extension of these methods to MIMO systems is generally possible by notational changes, e.g. scalars to vectors etc. , the task of implementation is involved. The implementation aspects have not been adequately addressed in the literature possibly because the implementation is often problem specific and depends on many factors such as the hardware and software enviroment. Other issues which are equally important such as data structures, algorithm organization, checks, validation, testing, and user interface are frequently neglected while extension to the MIMO case is being discussed.

The following factors were taken into consideration in developing an identification procedure for this project. Most identification procedures are based on open-loop plant behavior. It has been shown [40] that these procedures may fail to properly identify the open-loop plant dynamics when the plant is operating in a closed-loop configuration. This is a serious concern for the flexible manipulator because it is not safe to run the manipulator in the open-loop configuration. Gustavsson et al. [40]

have shown that if the recursive prediction error method (RPEM) fails to identify the open-loop parameters when the plant is operating in a closed-loop, all other methods will fail. Therefore RPEM was chosen as the method to be extended to the MIMO case. Although an identified MIMO ARX model would have been adequate for the MIMO GPC algorithm, a deliberate decision was made to identify the MIMO ARMAX model with the restriction that the noise polynomial identification is implemented without the necessary spectral factorization algorithm. If the noise polynomials are not identified, the RPEM method reduces to the standard recursive least square (RLS) identification method.

Since the process of identification is equivalent to the linearization of the plant behavior about an operating point and the dynamics of the flexible manipulator are highly nonlinear and the operating point is changing, the identification procedure must be capable of following time-varying parameters.

Ljung and Gunnarson [62] conclude that under ideal conditions the Kalman filter would provide the best identification results. In reality, the parameter and output noise statistical properties are not known, therefore, some modifications of the RLS or LMS (Least Mean Square) methods are typically used. It is concluded that the main advantage of RLS over LMS is its ability to achieve faster good estimates in case of abrupt changes in the system parameters.

The choice of the forgetting factor for the identification procedure is well examined by Shook [90] and Rogers [84]. For this work, the DFF (Directional Forgetting Factor) was chosen and the results demonstrated that it performs well under many circumstances, for example, when the signal information content is low or when sudden changes in parameters occur.

The approach taken here is consistent with the MIMO GPC, i.e. the MIMO ARMAX is seen as a set of MISO systems, each of them described by the appropriate row of the MIMO ARMAX model. Before going into the MIMO RPEM implementation issues, the RPEM identification algorithm is given for the i-th row MISO subsystem with the DFF. It is assumed that the one-step ahead prediction equation is in the pseudolinear (or linear for ARX models) form, i.e.

$$y_i(t) = \hat{\theta}_i^T(t)\varphi_i(t) + e_i(t) \tag{5.135}$$

where $y_i(t)$, $\hat{\theta}_i(t)$, $\varphi_i(t)$, and $e_i(t)$ are the output, parameter vector, regressor vector, and prediction error, respectively, of the i-th MISO subsystem. The parameter and

regressor vectors are defined as follows

$$\begin{aligned}\widehat{\theta}^T = [\ & \widehat{a}_{i1,1} \quad \widehat{a}_{i1,2} \quad \dots \quad \widehat{a}_{i1,n_{A_{i1}}} \\ & \widehat{a}_{i2,1} \quad \widehat{a}_{i2,2} \quad \dots \quad \widehat{a}_{in,n_{A_{in}}} \\ & \widehat{b}_{i1,1} \quad \widehat{b}_{i1,2} \quad \dots \quad \widehat{b}_{im,n_{B_{im}}} \\ & \widehat{c}_{ii,1} \quad \widehat{c}_{ii,2} \quad \dots \quad \widehat{c}_{ii,n_{C_{ii}}} \ ]\end{aligned} \tag{5.136}$$

$$\begin{aligned}\varphi^T = [\ & -y_1(t-1) \quad -y_1(t-2) \quad \dots \quad -y_1(t-n_{A_{i1}}) \\ & -y_2(t-1) \quad -y_2(t-2) \quad \dots \quad -y_n(t-n_{A_{in}}) \\ & u_1(t-1) \quad u_1(t-2) \quad \dots \quad u_m(t-n_{B_{im}}) \\ & \epsilon_i(t-1) \quad \epsilon_i(t-2) \quad \dots \quad \epsilon_i(t-n_{C_{ii}}) \ ]\end{aligned} \tag{5.137}$$

In the above definitions, note that the noise effects have been lumped into the diagonal noise polynomials $C_{ii}(q^{-1})$. In the following identification sequence of equations, $\mathbf{P}_i$ and $\psi_i(t)$ denote the covariance matrix and the gradient vector, respectively, for the i-th row subsystem. The definition of remaining variables used to implement the DFF can be found in [58].

$$\epsilon_i(t|t-1) = y_i(t) - \widehat{\theta}_i^T(t|t-1)\varphi_i(t) \tag{5.138}$$

$$\zeta_i(t|t-1) = \psi_i^T(t)\mathbf{P}_i(t|t-1)\psi_i(t) \tag{5.139}$$

$$\widehat{\theta}_i(t+1|t) = \widehat{\theta}_i(t|t-1) + \frac{\mathbf{P}_i(t|t-1)\psi_i(t)}{1+\zeta_i(t|t-1)}\epsilon_i(t|t-1) \tag{5.140}$$

$$\epsilon_{iN}^2(t|t-1) = \frac{\epsilon_i^2(t|t-1)}{\sigma_i^2(1+\zeta_i(t|t-1))} \tag{5.141}$$

$$\lambda_i^{-1}(t+1|t) = 1 + (1+\rho_i)\left\{\ln\left[1+\zeta_i(t|t-1)\right] + \frac{\zeta_i(t|t-1)}{1+\zeta_i(t|t-1)}\left[\epsilon_{iN}^2(t|t-1) - 1\right]\right\} \tag{5.142}$$

$$\gamma_i(t) = \lambda_i(t+1|t) - \frac{1-\lambda_i(t+1|t)}{\zeta_i(t|t-1)} \tag{5.143}$$

$$\mathbf{P}_i(t+1|t) = \mathbf{P}_i(t|t-1) - \frac{\mathbf{P}_i(t|t-1)\psi_i(t)\psi_i^T(t)\mathbf{P}_i(t|t-1)}{\frac{1}{\gamma_i(t)} + \psi_i^T(t)\mathbf{P}_i(t|t-1)\psi_i(t)} \tag{5.144}$$

$$\epsilon_i(t) = \frac{1}{\widehat{C}_{ii}}\left(\sum_{j=1}^{n}\sum_{k=1}^{n_{A_{ij}}} a_{ij,k}y_j(t-k) - \sum_{j=1}^{m}\sum_{k=1}^{n_{B_{ij}}} b_{ij,k}u_j(t-k)\right) \tag{5.145}$$

$$\psi_i(t+1) = \frac{1}{\widehat{C}_{ii}}\varphi_i(t+1) \tag{5.146}$$

Identification $T(q^{-1})$ Filter.

Since identification methods using positional data suffer from bias of the identified parameters, it is usual to use the incremental version of the RPEM. However, this results in identified parameters describing behavior of the system in higher frequencies which is not desirable. Wahlberg and Ljung [104] recommend additional signal prefiltering because the differencing suppresses the low frequency signal components while the the high frequency components are amplified. Clarke [23] recommends that all signals are filtered through $\Delta(q^{-1})/T(q^{-1})$ where $T(q^{-1})$ is the identification T filter and should be of order larger or equal to the order of the identified system so that the character of prefiltering is a band-pass filter with a frequency range within the actuator bandwidth. Lambert [59], however, identified a 4-th order system with $deg(T(q^{-1})) = 2$ and claimed that higher order of $T(q^{-1})$ is not necessary. It must be pointed out that the $T(q^{-1})$ filter for GPC and $T(q^{-1})$ filter for RPEM do not have to, and frequently should not, be identical.

Shook [90] presents a better approach to identification data prefiltering to obtain a good identification model by implementing an adaptive filter which results in a better model than the one chosen in ad hoc fashion. The only shortcoming is that it requires solution of Diophantine equations and spectral factorization. This is not such a significant computational burden in cases where the predictions are calculated on the base of the Diophantine equations. However, when the predictions are calculated via successive substitutions then the Diophantine equations for the MIMO system may represent an unacceptable computer load. It is obvious, though, that this approach should be implemented in the future even if it means lower maximum sampling frequency because a better identified model may give better response at lower sampling frequency than a poor model based on a higher sampling frequency.

The RPEM was implemented as an incremental method with $T(q^{-1})$ filters specified by the user. Currently, the maximum order of the $T(q^{-1})$ filters is 6. There is a $T(q^{-1})$ filter for each input and output and the user can modify their coefficients in the *setupvar* file. Note, that if one would like to perform the identification in its positional form, then all $T(q^{-1})$ filters must contain $\Delta(q^{-1})$ term.

Implementation Aspects.

Equations 5.135-5.146 represent an integration of MISO RPEM for ARMAX model with DFF. Such a written algorithm gives the illusion of completeness, but in reality it is only a small part of the identification algorithm and does not say anything about possible numerical problems.

To guarantee numerical stability of the identification algorithm, it is necessary to ensure that the covariance matrix $\mathbf{P}_i$ remains positive definite. A typical procedure

is to use the Bierman's UD factorization method [16] which maintains the covariance matrix in the form of $\mathbf{P}_i = \mathbf{U}_i\mathbf{D}_i\mathbf{U}_i^T$ and guarantees the positive definiteness. The actual implementation of this algorithm is according to Thornton and Bierman [99]. However, to obtain efficient integration of the UD factorization algorithm and the DFF, the UD algorithm was modified so that no calculations are repeated and tests on sufficient persistent excitation are built in as well.

Note, that if the noise polynomials are not estimated then the gradient vectors, $\psi_i(t)$, become regressor vectors, $\varphi_i(t)$, and the entire algorithm becomes MIMO RLS with the underlying MIMO ARX model.

When dealing with the MIMO identification, it is necessary to implement the algorithm efficiently and with a reasonable degree of flexibility. The efficiency is needed so that the identification algorithm is fast and correct. The flexibility means that a user can easily change the structure of the identified model without any need for reprogramming and recompilation of the program and can rerun the experiment with different identified models within a short time.

The MIMO RPEM was implemented such that the user can change the orders of the identified polynomials and orders of delay for each row subsystem. For the purpose of the MIMO GPC the delay orders should be set to zero so that the models used by the MIMO RPEM and MIMO GPC are consistent because the delays are then in the form of very small values of leading coefficients of B_{ij}. Currently, the entire program reads many of its input parameters from a file named *setupvar*. The model structure is a 4×7 integer matrix of the following structure

$$\begin{array}{cccc:cc:c} n_{A_{11}} & n_{A_{12}} & n_{A_{13}} & n_{A_{14}} & n_{B_{11}} & n_{B_{12}} & n_{C_{11}} \\ n_{A_{21}} & n_{A_{22}} & n_{A_{23}} & n_{A_{24}} & n_{B_{21}} & n_{B_{22}} & n_{C_{22}} \\ n_{A_{31}} & n_{A_{32}} & n_{A_{33}} & n_{A_{34}} & n_{B_{31}} & n_{B_{32}} & n_{C_{33}} \\ n_{A_{41}} & n_{A_{42}} & n_{A_{43}} & n_{A_{44}} & n_{B_{41}} & n_{B_{42}} & n_{C_{44}} \end{array} \tag{5.147}$$

with the delay information matrix given by

$$\begin{array}{cc} \Delta_{11} & \Delta_{12} \\ \Delta_{21} & \Delta_{22} \\ \Delta_{31} & \Delta_{32} \\ \Delta_{41} & \Delta_{42} \end{array} \tag{5.148}$$

and the vector of initial values of the forgetting factors

$$\beta_1 \quad \beta_2 \quad \beta_3 \quad \beta_4 \tag{5.149}$$

The dimension of these matrices $n \times (n + m + 1)$ and $n \times m$ are not fixed. There are two key variables Y and U currently set to $Y = 4$ and $U = 2$ in the entire program which define the dimensions of the system to be identified and controlled. By changing these two variables, any system with $n \leq Y$ and $m \leq U$ can be accommodated. The second integer matrix describing minimum existing delays is also influenced by the two variables Y and U. The delay matrix was introduced to the MIMO RPEM identification to minimize the number of identified parameters in case there are inherent delays in the system. There is also one line describing the vector of initial values of the forgetting factor which is not as important with the DFF implementation because the forgetting factor is variable and controlled by information contents of input and output signals.

The MIMO RPEM like any other identification algorithm must go through its initialization part. All information that is needed is contained in the two variables Y and U, and the three arrays which are in the input file *setupvar*. The first array defining the orders of the identified polynomials describes not only the orders of particular polynomials but also couplings in the system. If a polynomial order is set to zero then it means that it will not be identified. For example, if we wanted to identify a 4×2 model without any output couplings, all $n_{A_{ij}}$ with $i \neq j$ should be set to zero.

The algorithm striving for efficiency must during its initialization phase prepare many arrays and variables which help to perform all identification operations with minimum overhead. As was mentioned in the case of the MIMO GPC, it is beneficial to store all vectors as row vectors because all operations can be performed via access through incremented pointers.

The algorithm must determine the number of identified parameters for a particular MISO subsystem which defines the dimensions of regressors $\varphi_i(t)$, gradients $\psi_i(t)$, parameter vectors $\widehat{\theta}_i(t)$, and covariance matrices $\mathbf{P}_i(t)$. Another part of the initialization algorithm must decide the maximum number of samples for each input, output, and residual error signals which must be periodically updated during each sampling interval. Each signal has its own offset which defines its location in $\varphi_i(t)$ and $\psi_i(t)$ where the latest sample should be stored to participate in the next identification step. Matrices containing orders of each identified polynomial and the address of its

first element must be properly initialized as well. These are just a few of many steps of the initialization section. The complete implementation of the entire initialization and identification can be found in the C program source code.

For the purpose of initialization of the regressor vector, in case of a real experiment, the system is driven by a small control signal without identification and the GPC algorithm. Duration of this open-loop mode is determined by the maximum polynomial order of the system. Only then the initialized regressor and preset parameter vectors are employed in the identification algorithm.

After each identification step the regressor, gradient, and signal vectors are shifted and updated for the next identification step. The regressor, gradient, and parameter vectors are stored in contiguous row vectors so that the UD algorithm and parameter update using the Kalman filer can be carried out efficiently.

Chapter 6

Simulations

6.1 Introduction

The purpose of this chapter is to test the MIMO GPC and MIMO RPEM algorithms via simulations. The essential advantage of the simulation runs is that they are carried out in a well controlled environment, e.g. noise, plant/model mismatch etc. When a complex control algorithm is implemented, it is almost impossible to prove that it is completely correct. Control algorithms have an additional problem related to the complexity of the controlled plant because exact behavior of the controlled plant under a complex control algorithm is not known and it is therefore difficult to conclude if the tested algorithm is correct. Since exhaustive testing is impossible, a typical procedure is to test the algorithm with simple plants and if the performance is satisfactory then it is assumed that the algorithm is correct. Such an approach is well justified in the case of SISO algorithms. The situation is different for MIMO algorithms when the program may have some hidden errors which can be demonstrated only under specific conditions. In the case of the MIMO GPC and MIMO RPEM the tests are influenced by the interactions between the control algorithm and the identification procedure, the increased number of tuning knobs, and parameter initialization.

In short, simulations serve at least two purposes: testing the control algorithm performance under well defined conditions and discovering implementation errors. The same is valid for the identification procedure and the combination of the MIMO GPC and MIMO RPEM.

This chapter is organized as follows. Section 6.2 deals with smooth set-point transitions. Different simulation runs with a nonsquare output decoupled plant are presented in section 6.3. A model with couplings obtained from the flexible manipula-

tor moving in the horizontal plane is used for different simulation runs in section 6.4. Conclusions and observations from the presented simulation runs are described in section 6.5. All simulations were programmed in C programming language. The same code is used in the actual experiments described in the next chapter.

6.2 Smooth Set-point Transition

When, initially, the GPC code was tested without identification (no plant model mismatch) and without any noise, it was observed that the response to the future set-point step changes moved initially in the opposite direction to the set-point change. This non-minimum phase like behavior is undesirable for the flexible manipulator. It happens whenever $1 < NU < N2$. There are several ways to counteract this behavior: a sluggish response with $NU = 1$ or an almost dead-beat response with $NU = N2$ represent two possibilities. Another remedy is a sufficiently large amplitude of the control weighting coefficient λ_u. However, restrictions to $NU = 1$ or $NU = N2$ as well as the large control weighting coefficient are not always acceptable solutions. Figure 6.1 shows this situation for a double integrator plant with $NU = 2$ controlled by a deterministic SISO GPC.

When the situation was empirically analyzed from the simulation runs, it was observed that at the time instant when the future step change in the set-point is $N2$ steps ahead of the current time instant then the following situation happens. The vector of the difference between the future set-point and the predicted response has only the last element nonzero and the $(\mathbf{G}^T\mathbf{G}+\lambda_u\mathbf{I})^{-1}\mathbf{G}^T$ matrix has a relatively large and negative last element in its first row. This causes the first element of the control increment vector to be negative and drives the controlled system in the opposite direction to the set-point change, i.e.

$$\Delta\mathbf{u}(t) = (\mathbf{G}_T\mathbf{G} + \lambda_u\mathbf{I})^{-1}\,\mathbf{G}^T(\mathbf{w}-\mathbf{p}) = \begin{bmatrix} & & -k \\ & & \\ & & \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -k \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \tag{6.1}$$

The blank space in the above matrix indicates that all the other elements of this matrix are extremely small. If the entire control increment vector were used, instead of only its first component, the output would end up at the desired output level after

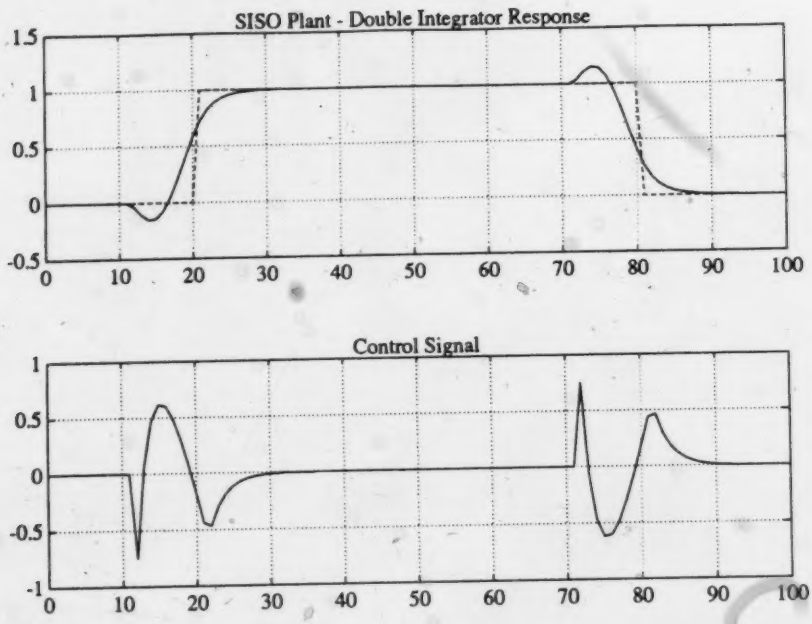


Figure 6.1: Output and control signals for a double integrator plant with step changes in set-point, $NU = 2$.

$N2$ steps.

Intuitively, if the $(\mathbf{w} - \mathbf{p})$ vector (the vectors have been defined in section 5.2) had all its elements sufficiently positive or the change in the future values of $(\mathbf{w} - \mathbf{p})$ were not so abrupt, the control increment vector should be expected to be positive and the undershoot should not occur. Therefore, the desired properties of the smooth set-point transition are a) $w(t) = p(t)$, b) $w(t + N2)$ is equal to the step change, and c)$w(t + j)$ gradually changes from $w(t)$ to $w(t + N2)$ over the output horizon.

In the study reported here the set-point vector $\mathbf{w}$ is altered sinusoidally. This is convenient for the motion of the flexible manipulator because $w'(t) = w'(t+N2) = 0$, which means smaller excitation of all modal frequencies. Simulations show a significant improvement when this sinusoidal transition is used. The sinusoidal transition has the following form

$$w(t+j) = w(t)+\frac{1}{2}\sin\left(-\frac{\pi}{2} + i\frac{\pi}{N2}\right)\{w(t + N_2) - w(t)\} \quad \text{for} \quad i = 1, 2, \ldots, (N2-1) \tag{6.2}$$

When the sinusoidal set-point transition is implemented, the timing requirements are important. Whenever a step change in the set-point appears for the first

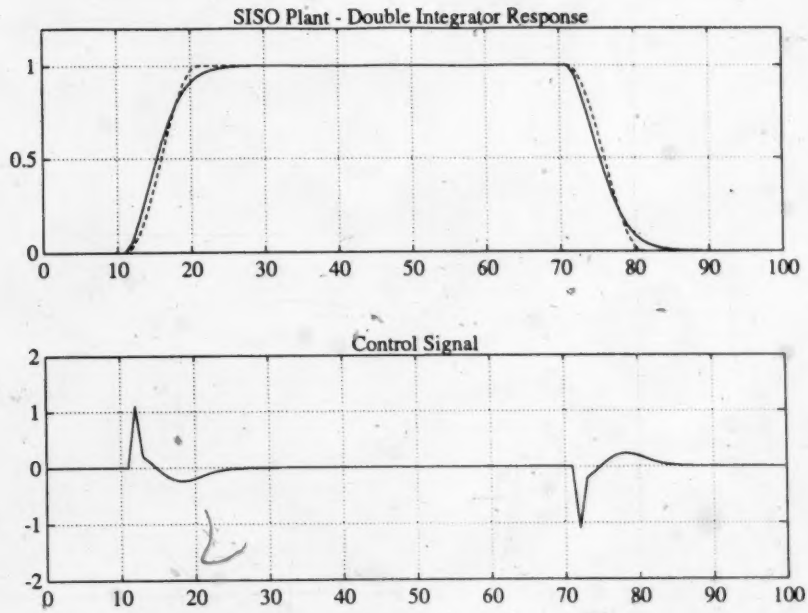


Figure 6.2: Output and control signals for a double integrator plant with sinusoidal transitions in set-point, $NU = 2$.

time in the output horizon of the **w** vector, the entire **w** vector must be rebuilt using the equation 6.2. Since the calculations involve a trigonometric function, their timing requirements for a larger $N2$ may exceed the time allocated to this set-point update operation. Therefore, a sinusoidal array is built during the initialization phase of the GPC algorithm so that the sinusoidal calculations according to equation 6.2 are reduced to floating point multiplications. Thus the time needed for the set-point recalculation is minimized. Figure 6.2 shows the beneficial effect of such a transition.

6.3 Output Decoupled Nonsquare Plant

Simulation results are presented for an output decoupled single-input two-output plant in this section. Although a standard approach to simulation studies includes effects of additive noise on the closed-loop performance and noise due to quantization effects, only the latter was used in the simulations described in this chapter. The omission of additive noise is motivated by the desire to investigate correctness and performance of the MIMO GPC and MIMO RPEM under well controlled conditions.

The experimental results in the next chapter will demonstrate how well GPC performs under real life conditions which include noise.

The main purpose of the simulation runs presented in this and the following section is to test different features of the MIMO GPC for nonsquare systems with $m < n$. This section uses an output decoupled system which consists of a double integrator with a gain in parallel with a very lightly damped second order system with a gain. The lightly damped system is described by $\zeta = 0.01$ and f_n=3 Hz. A sampling period $h = 0.01$ sec was used. The complete system can be either described by

$$y_1(t) = G_{11}(q^{-1})u_1(t) = \frac{0.0531(q^{-1} + q^{-2})}{1 - 1.9609q^{-1} + 0.9962q^{-2}}u_1(t) \tag{6.3}$$

$$y_2(t) = G_{21}(q^{-1})u_1(t) = \frac{0.05(q^{-1} + q^{-2})}{1 - 2q^{-1} + q^{-2}}u_1(t) \tag{6.4}$$

or as

$$\begin{pmatrix} A_{11}(q^{-1}) & A_{12}(q^{-1}) \\ A_{21}(q^{-1}) & A_{22}(q^{-1}) \end{pmatrix} \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} B_{11}(q^{-1}) & B_{12}(q^{-1}) \\ B_{21}(q^{-1}) & B_{22}(q^{-1}) \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} \tag{6.5}$$

with

$$A_{11}(q^{-1}) = 1 - 1.9609q^{-1} + 0.9962q^{-2} \tag{6.6}$$

$$A_{12}(q^{-1}) = 0 \tag{6.7}$$

$$A_{21}(q^{-1}) = 0 \tag{6.8}$$

$$A_{22}(q^{-1}) = 1 - 2q^{-1} + q^{-2} \tag{6.9}$$

$$B_{11}(q^{-1}) = 0.0531(q^{-1} + q^{-2}) \tag{6.10}$$

$$B_{12}(q^{-1}) = 0 \tag{6.11}$$

$$B_{21}(q^{-1}) = 0.05(q^{-1} + q^{-2}) \tag{6.12}$$

$$B_{22}(q^{-1}) = 0 \tag{6.13}$$

This output decoupled system was chosen, first, because it is a challenging plant to control and, second, it can be seen as a coarse approximation of the flexible manipulator moving only in the horizontal plane. The double integrator is a good approximation of the motor dynamics. The lightly damped second order system can be seen as an approximation of the flexible tip motion due to the first link modal frequency.

Simulation Run 1

Figure 6.3 shows the response and the control signal for a fully deterministic and exact case with $\lambda_{y1_{11}} = 1$, $\kappa_1 = 1$, $\lambda_{y2_{11}} = 1$, $\kappa_2 = 1$, $N1_1 = 1$, $N2_1 = 10$, $N1_2 = 1$, $N2_2 = 10$, $NU_1 = 2$, and with the active smooth set-point transition. This ideal case demonstrates that if the control requirements are not contradictory then the MIMO GPC can handle nonsquare systems. Note, that $NU_1 = 2$ was chosen as the initial choice of the control horizon because the double integrator as well as the lightly damped system represent systems with two badly located poles. Although altogether the entire plant has four badly located poles, $NU_1 = 2$ performs reasonably well. In all Figures in this chapter with the title "Outputs and Set-point" the solid line is used for the set-point and the double integrator, and the dashed line for the lightly damped system.

Simulation Run 2

Following the generally accepted guideline that the control horizon is made equal to the number of the badly located poles in the system, a larger control horizon $NU_1 = 4$ was chosen. Figure 6.4 shows the simulation results. The oscillatory behavior is damped but faster than in the previous simulation run. The control signal is more aggressive and the transitional behavior of the double integrator exhibits non-smooth motion because the MIMO GPC tries to minimize the oscillatory behavior of the lightly damped system at the same time. As the double integrator approaches the set-point level, the prediction error of the lightly damped system starts to dominate and the control signal is mostly aimed at minimizing this error which, in turn, has influence on the double integrator output.

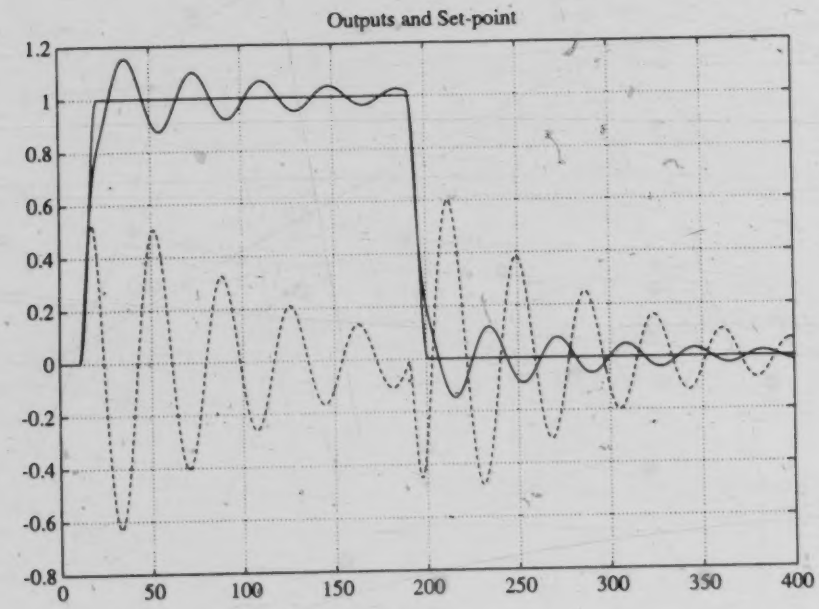
Simulation Run 3

Figure 6.5 shows the results for $\lambda_{y1_{11}} = 10$, $\kappa_1 = 1$, $\lambda_{y2_{11}} = 1$, and $\kappa_2 = 1$. Comparing these results with those in Figure 6.3 note that the prediction error of the lightly damped system has a larger weight than the prediction error of the double integrator. Such a choice leads to smaller oscillatory behavior of the lightly damped system and to more oscillatory response of the double integrator.

Simulation Run 4

The effects of the plant/model mismatch are considered next. The plant model used by the MIMO GPC was changed to $A_{11}(q^{-1}) = 1 - 1.9609q^{-1}$, to give an unstable model of smaller order. Several T-filters were tried but none of them resulted in acceptable closed-loop behavior. Figure 6.6 show the system response for the same settings as for Figure 6.3, i.e. $NU_1 = 2$, $N1_1 = N1_2 = 1$, and $N2_1 = N2_2 = 10$. Note the control signal amplitude clipping feature at ± 10V.

Simulation Run 5



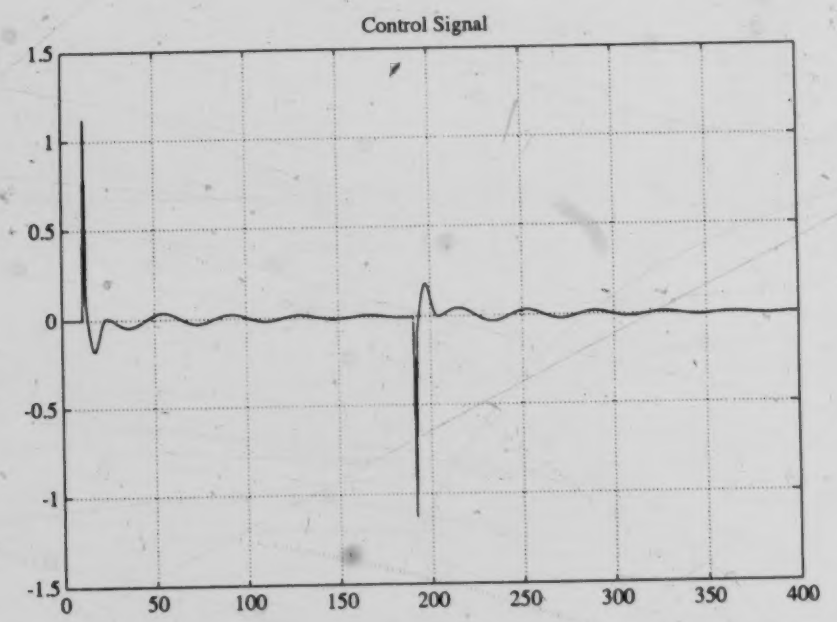


Figure 6.3: Simulation Run 1.

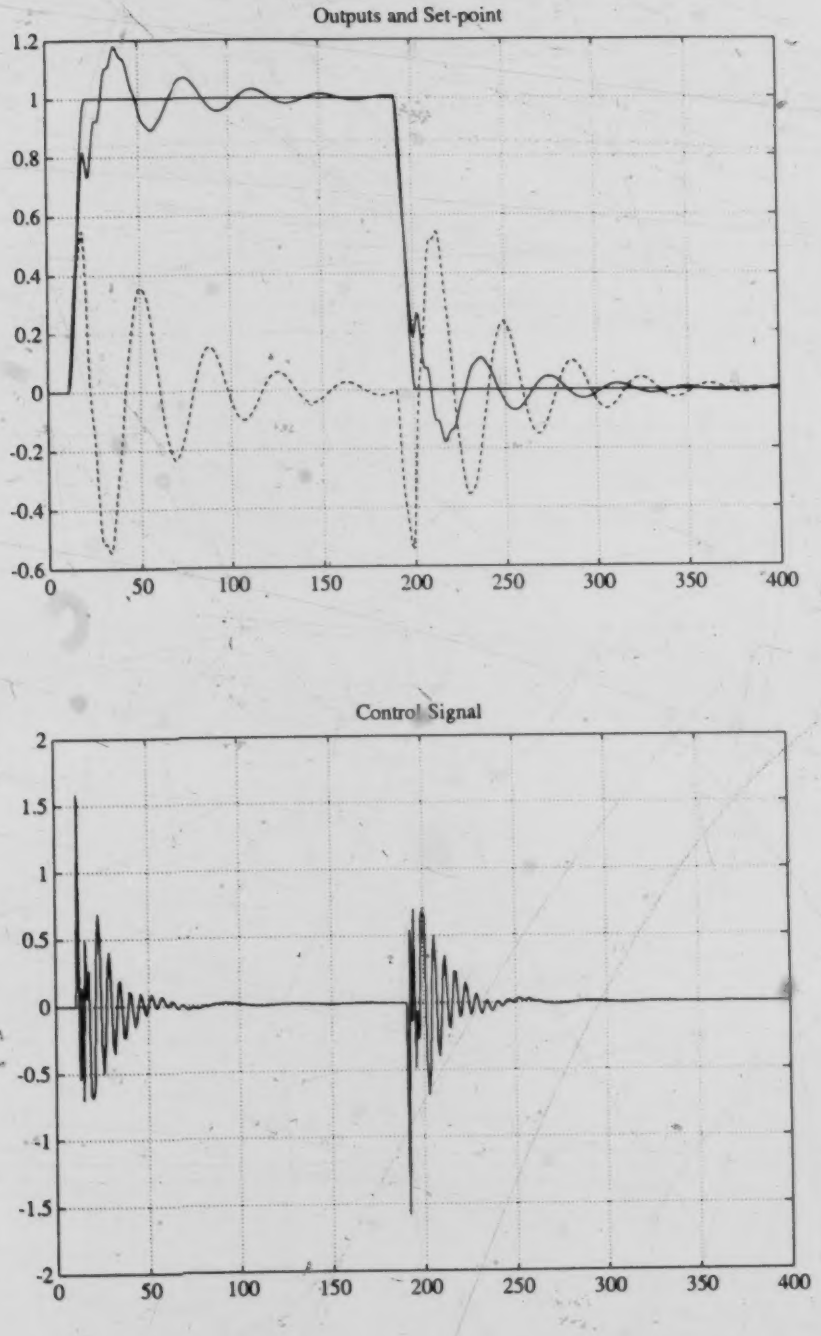


Figure 6.4: Simulation Run 2.

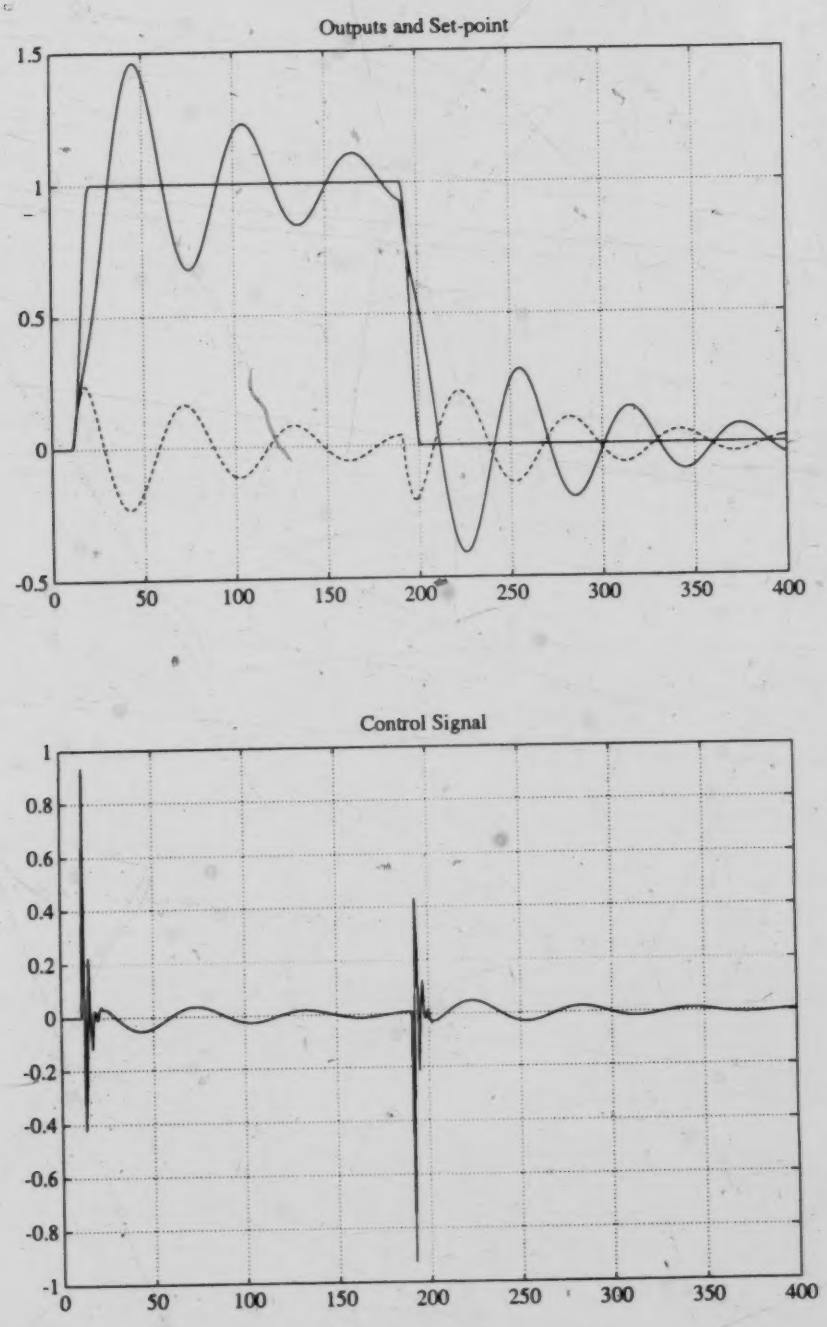


Figure 6.5: Simulation Run 3.

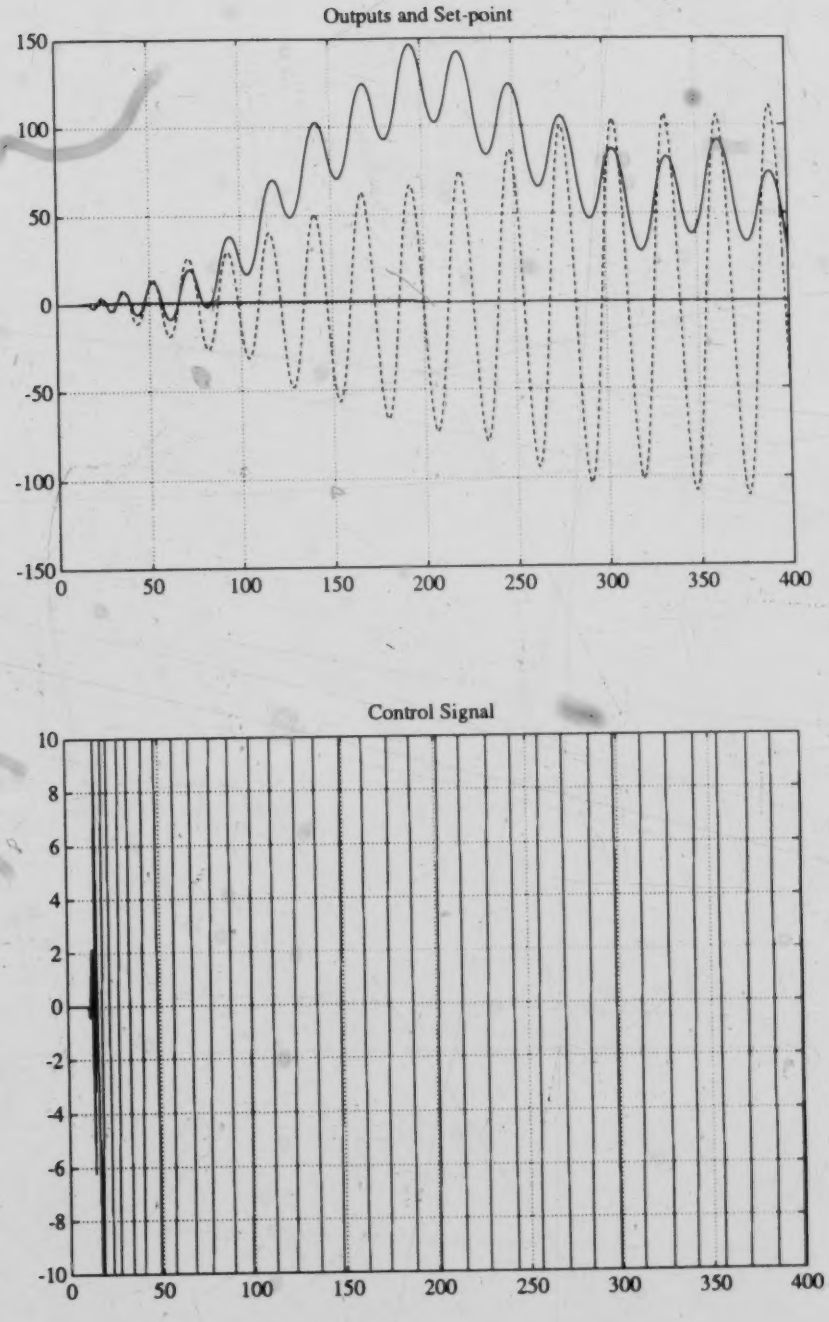


Figure 6.6: Simulation Run 4.

Figure 6.7 shows dramatic improvement when the following weighting coefficients were used: $\lambda_{y1_{11}} = 1$, $\kappa_1 = 0.5$, $\lambda_{y2_{11}} = 2$, and $\kappa_2 = 1.5$. This improvement is mostly because of very small weight of the slightly damped system prediction error. However, this prediction error was not completely zeroed and the oscillatory response of the entire system shows its impact.

Simulation Run 6

Figure 6.8 shows the results when the adaptive version of the MIMO GPC is used. We use the same plant mode as in the previous run, with $\lambda_{y1_{11}} = 1$, $\kappa_1 = 1$, $\lambda_{y2_{11}} = 1$, and $\kappa_2 = 1$. The quantization effects on input and output signals are now included. The MIMO RPEM is initialized exactly as the model used by the MIMO GPC in the previous simulation run, i.e. the MIMO RPEM uses only a first order model to describe the lightly damped system with the initialization $A_{11}(q^{-1}) = 1 - 1.9609q^{-1}$. The MIMO RPEM is used in its positional version for this simulation run. Although at the beginning of the simulation run the responses look promising, eventually the identified parameter $A_{11,1}$ changes its value dramatically, i.e. it alternates and at certain instants depicts an unstable first order plant. The plant instability is not a problem for the MIMO GPC, it is the instability of the the parameter $A_{11,1}$ that is the cause of this poor performance.

Simulation Run 7

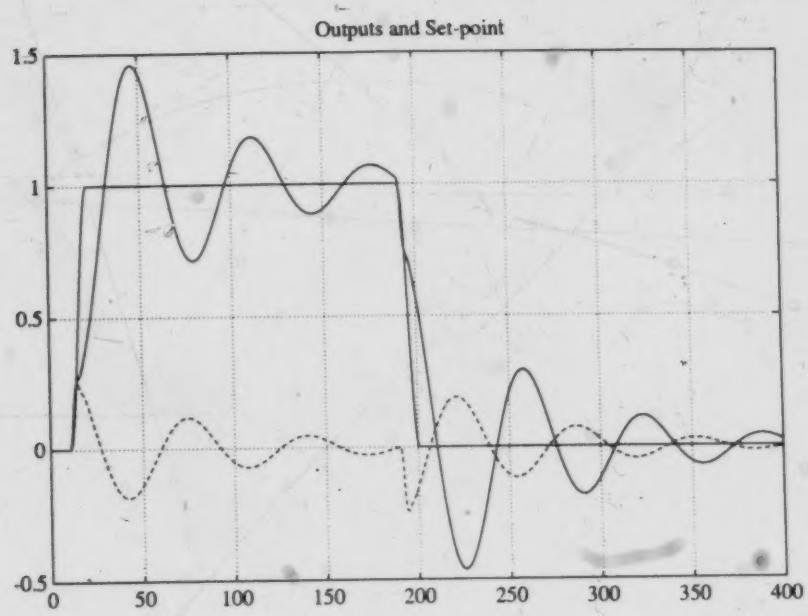
Figure 6.9 repeats the simulation run from Figure 6.8 with additionally applied T-filters in the form $T(q^{-1}) = (1 - 0.05q^{-1})^2$ for both the GPC and RPEM. Their application shows significant improvement not only in the response but also in the identified parameter $A_{11,1}$. The MIMO RPEM is used in its incremental form. This choice of the T-filter allows a wide frequency range of the control signal and, at the same time, the effects of high frequency dynamics on the identified model are suppressed. The overall performance is very satisfactory.

Simulation Run 8

To examine the effects of omitting the GPC T-filter, the previous simulation run was repeated with only the RPEM T-filter retained. The incremental version of the MIMO RPEM was used. Figure 6.10 shows the result of a simulation run. Note that the identified parameter $A_{11,1}$ stays in the neighborhood of one value, but the response is not satisfactory. Comparison with the previous simulation run clearly demonstrates the desirable effects of including the GPC T-filter.

Simulation Run 9

Figure 6.11 shows the system behavior when the plant model is overparametrized and



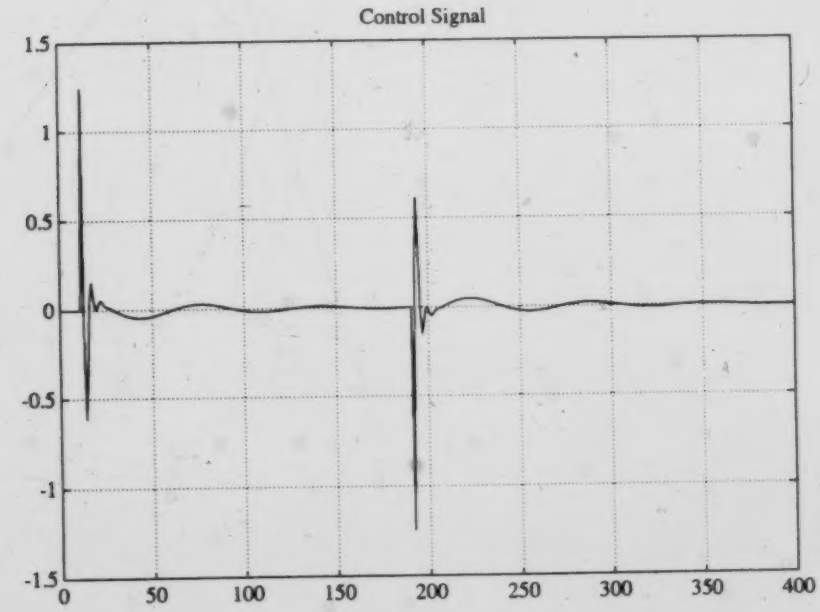


Figure 6.7: Simulation Run 5.

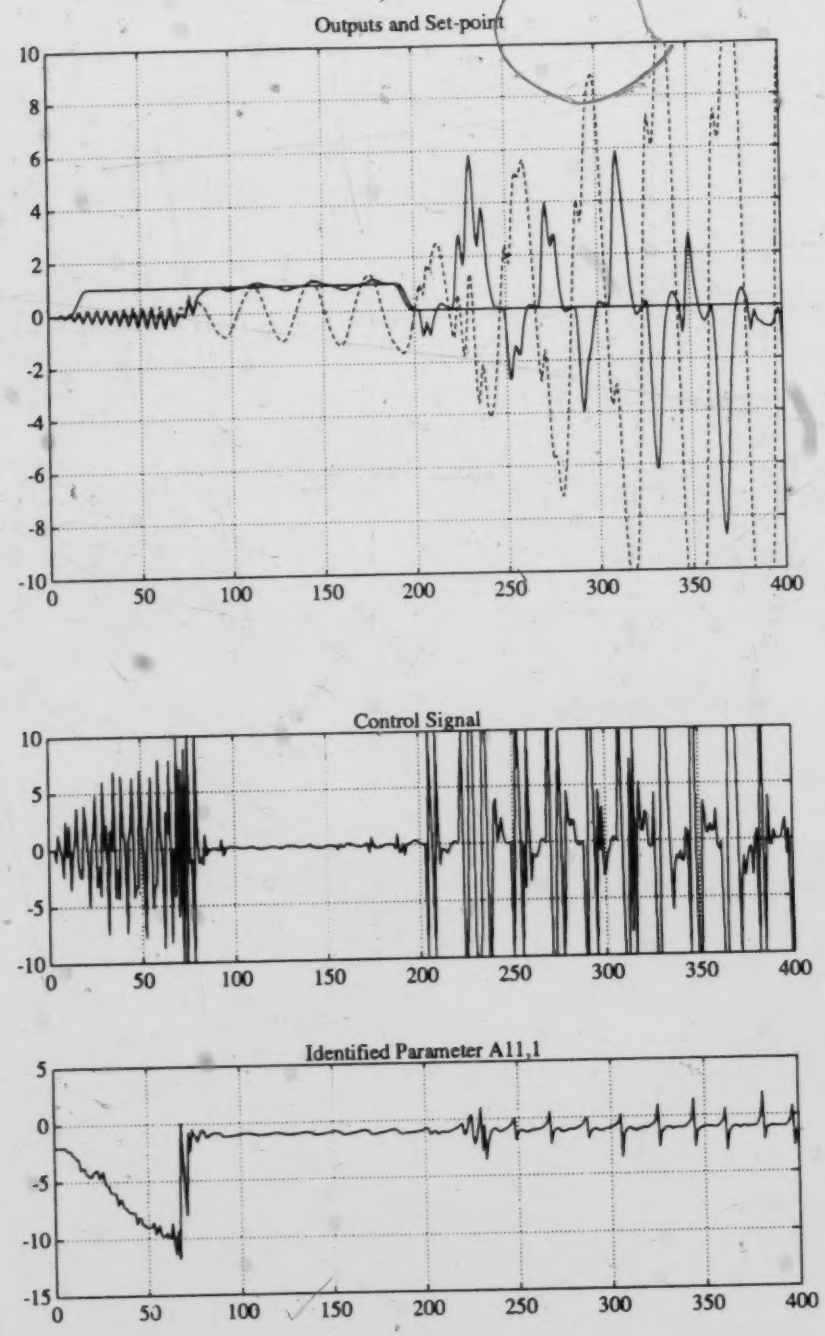


Figure 6.8: Simulation Run 6.

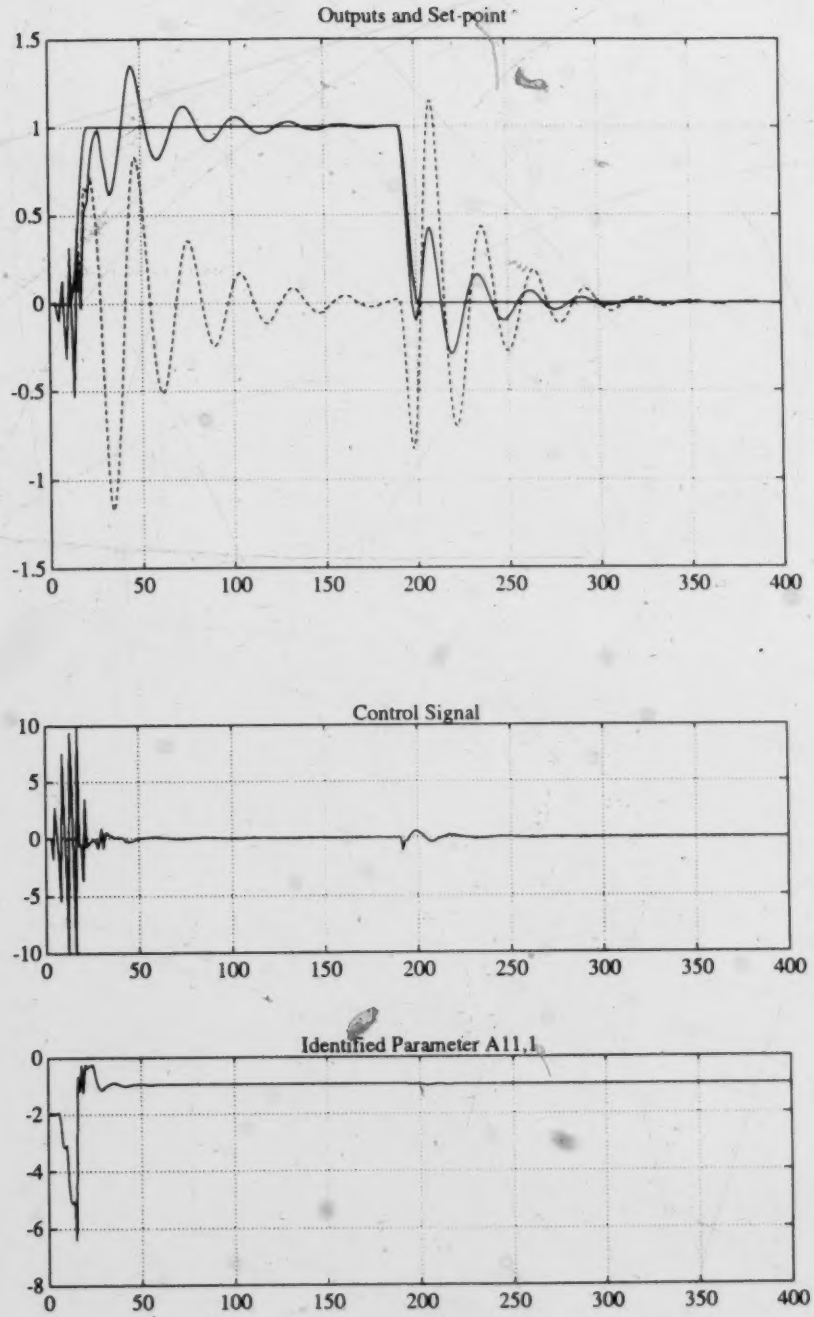


Figure 6.9: Simulation Run 7.

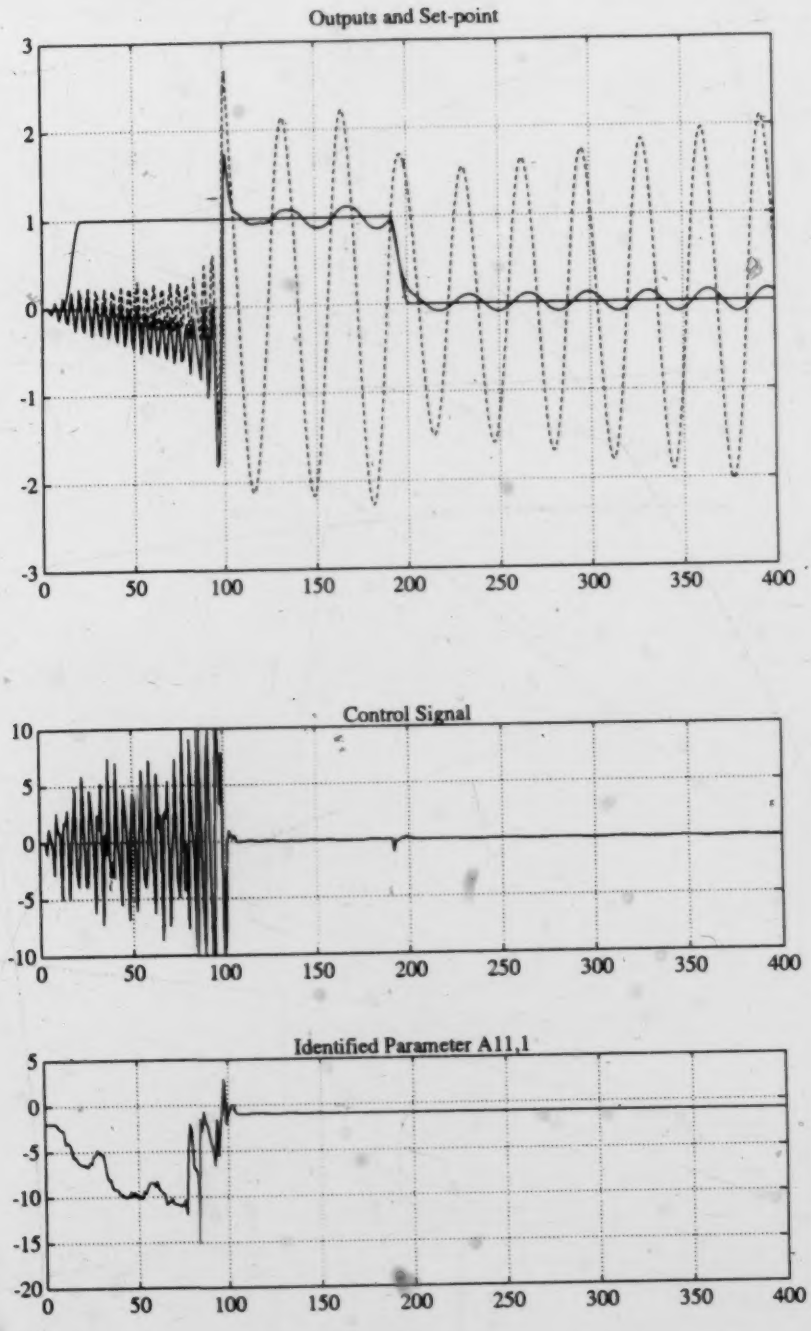


Figure 6.10: Simulation Run 8.

the initial values of the model parameters are only approximate, i.e.

$$A_{11}(q^{-1}) = 1 - 1.6q^{-1} + 0.8q^{-2} + 0.0q^{3} \quad (6.14)$$

$$A_{22}(q^{-1}) = 1 - 1.9q^{-1} + 0.9q^{-2} + 0.01q^{3} \quad (6.15)$$

$$B_{11}(q^{-1}) = 0.1q^{-1} + 0.1q^{-2} + 0.1q^{3} \quad (6.16)$$

$$B_{21}(q^{-1}) = 0.1q^{-1} + 0.1q^{-2} + 0.1q^{3} \quad (6.17)$$

The MIMO RPEM is used in its positional form and without any T-filter. Note the error in the double integrator response. This error was also observed when the MIMO RPEM was tried in its incremental form. When the double integrator was parameterized correctly, i.e. $\deg(A_{22}) = \deg(B_{21}) = 2$, the error disappeared.

6.4 Output Coupled Nonsquare Plant

In this section, we consider a single-input two-output plant with output coupling. The model is derived by identifying the parameters of the flexible manipulator moving only in the horizontal plane. This was done to ensure that the model used will be realistic and show strong output coupling effects. It has the following form

$$A_{11}(q^{-1}) = 1 - 0.649q^{-1} - 0.729q^{-2} + 0.232q^{-3} \quad (6.18)$$

$$A_{12}(q^{-1}) = -3.36q^{-1} + 3.13q^{-2} + 4.99q^{-3} - 4.76q^{-4} \quad (6.19)$$

$$B_{11}(q^{-1}) = -0.0979q^{-1} - 0.7522q^{-2} + 0.597q^{-3} + 0.1697q^{-4} + 1.1213q^{-5} - 1.0362q^{-6} \quad (6.20)$$

$$A_{21}(q^{-1}) = +0.0041q^{-1} - 0.0032q^{-2} \quad (6.21)$$

$$A_{22}(q^{-1}) = 1 - 1.8644q^{-1} + .8648q^{-2} \quad (6.22)$$

$$B_{21}(q^{-1}) = -0.0379q^{-1} + 0.0748q^{-2} + 0.0386q^{-3} - 0.0242q^{-4} - 0.0276q^{-5} + 0.0131q^{-6} \quad (6.23)$$

The above model shows that the coupling from the motor position to the tip acceleration is much stronger than the coupling from the tip acceleration to the motor position.

Simulation Run 10

Figure 6.12 shows the behavior of the system when the model used by the MIMO GPC is the same as the model of the plant. The GPC tuning knobs are $N1_1 = N1_2 = 1$, $N2_1 = N2_2 = 10$, $NU_1 = 2$, $\lambda_{y1_{11}} = 1$, $\kappa_1 = 1$, $\lambda_{y2_{11}} = 1$, and $\kappa_2 = 1$. There is

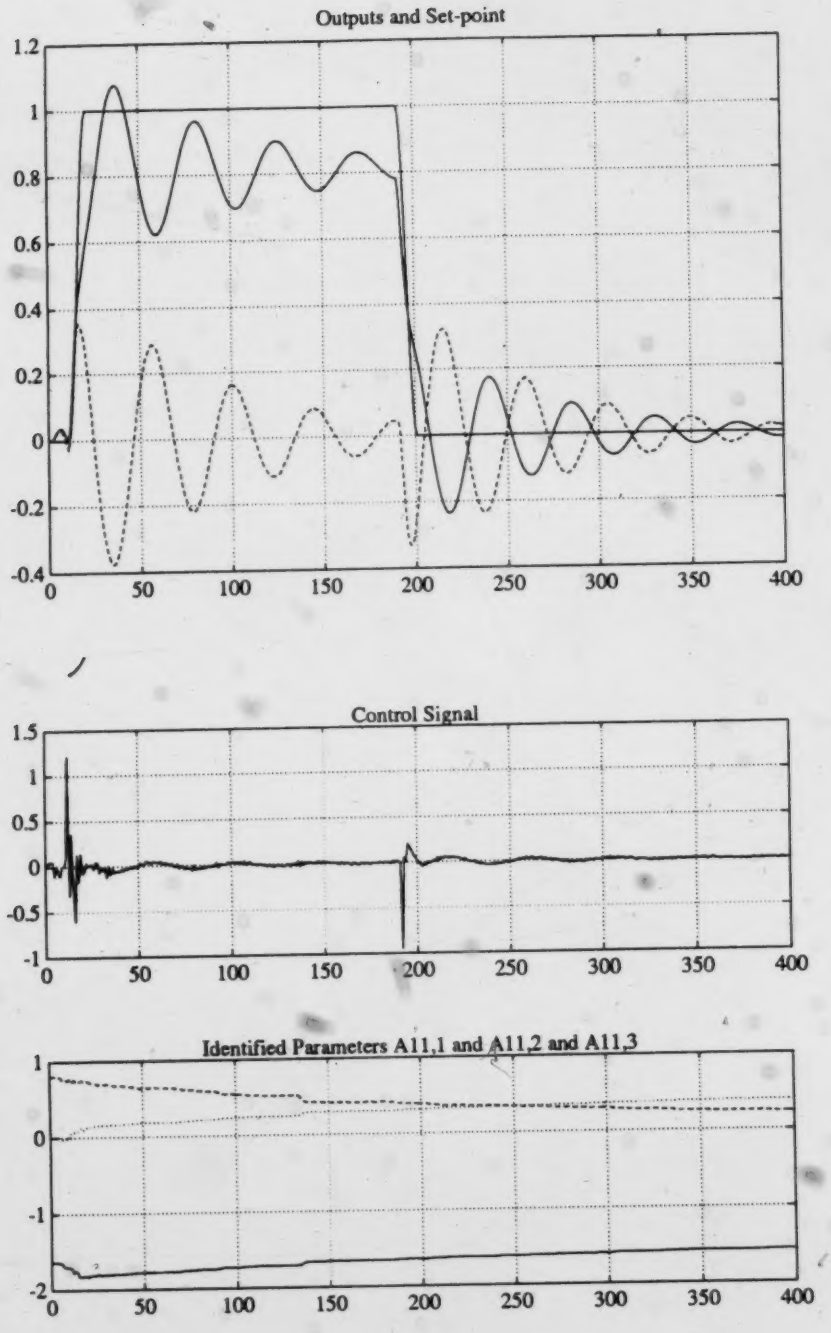


Figure 6.11: Simulation Run 9.

no T-filter. Although the dynamic performance is satisfactory, the motor position exhibits steady state error. Also, the system becomes unstable for any $NU_1 \neq 2$.

Simulation Run 11

To compensate for the steady state error, the simulation run was repeated with only one change in the output error weighting coefficient $\kappa_2 = 1.061$. As can be seen in Figure 6.13, the motor position is without the steady state error. Note the small nonzero value of the control signal when the motor position settles during the first half of this simulation run. This can be attributed to the coupling terms.

Simulation Run 12

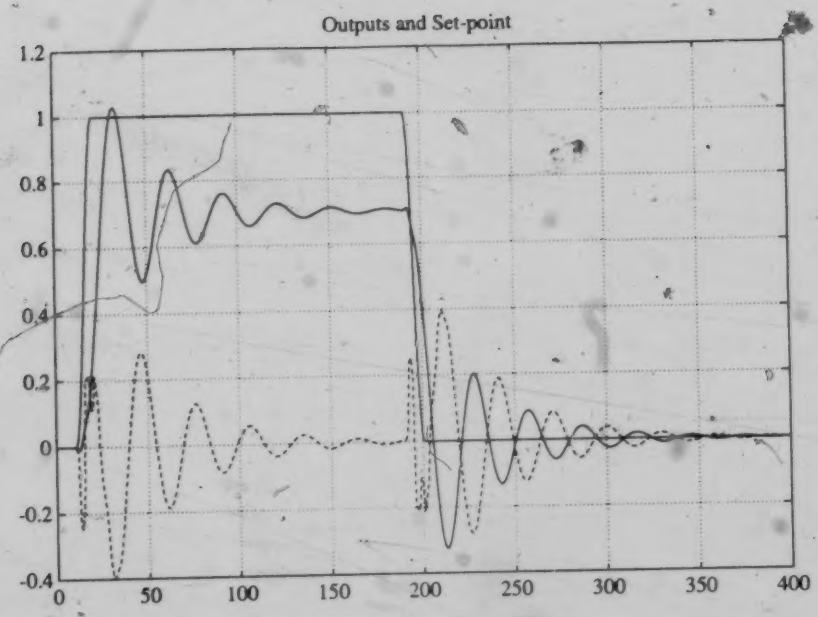
Figure 6.14 repeats the previous simulations but this time with the MIMO RPEM and exact parameter initialization. The noise due to the quantization of input and output signals is also in effect. Since the RPEM identification procedure provides slightly changed system parameters, the steady-state error is observed again. Figure 6.14 shows the identified first two coefficients of $A_{11}(q^{-1})$. Although not shown here, identification runs with underparameterized coupled terms are not satisfactory and lead frequently to unstable behavior. Overparameterization of the coupled plant, on the other hand, gives good results as long as the diagonal terms $A_{ii}(q^{-1})$ are overparameterized and the coupling terms $A_{ij}(q^{-1})$ are underparameterized. However, if overparameterization is used, it is necessary to increase the value of the control horizon, possibly because more poorly located poles appear as a result of the identification procedure.

Simulation Run 13

Figure 6.15 shows the scenario when the adaptive GPC with positional RPEM is used and only the diagonal terms are identified and overparameterized, i.e. $\deg(A_{11}) = \deg(A_{22}) = 6$, $\deg(A_{12}) = \deg(A_{21}) = 0$, and $\deg(B_{11}) = \deg(B_{21}) = 6$. While the acceleration is damped out well, the motor position response is rather sluggish and exhibits steady state error. This response was obtained with $NU_1 = 2$. Smaller values of NU_1 resulted in a smoother response and larger values of NU_1 resulted in a response with very abrupt behavior.

Simulation Run 14

Figure 6.16 uses the adaptive GPC with the incremental version of the MIMO RPEM. The GPC T-filter is $T(q^{-1}) = 1-0.9q^{-1}$ and the RPEM T-filter is $T(q^{-1}) = 1-0.5q^{-1}$. The system parameterization is characterized by $\deg(A_{11}) = 4$, $\deg(A_{22}) = 2$, $\deg(A_{12}) = \deg(A_{21}) = 0$, and $\deg(B_{11}) = \deg(B_{21}) = 6$. The parameter initialization is not exact. The initial delay in the response of the motor shaft can be attributed to the parameter identification. The transient response improved during the second



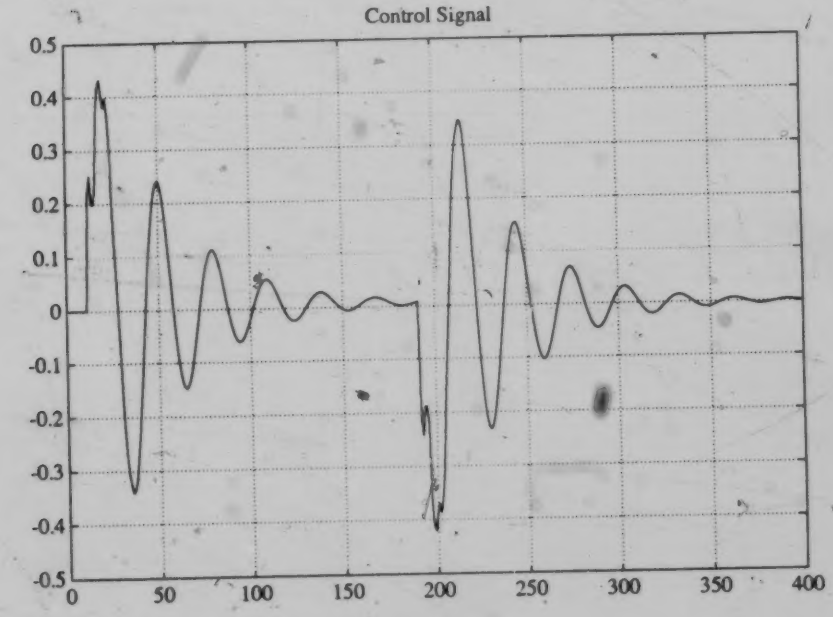
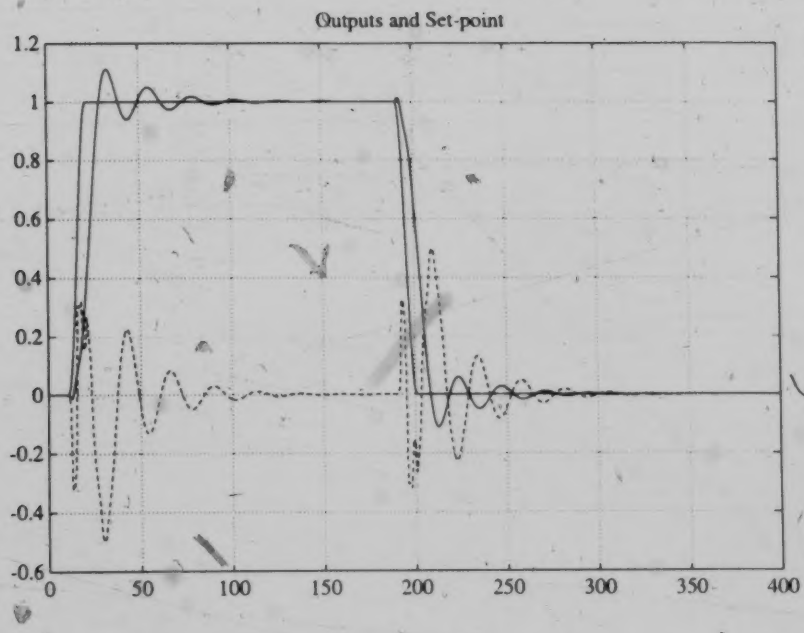


Figure 6.12: Simulation Run 10.



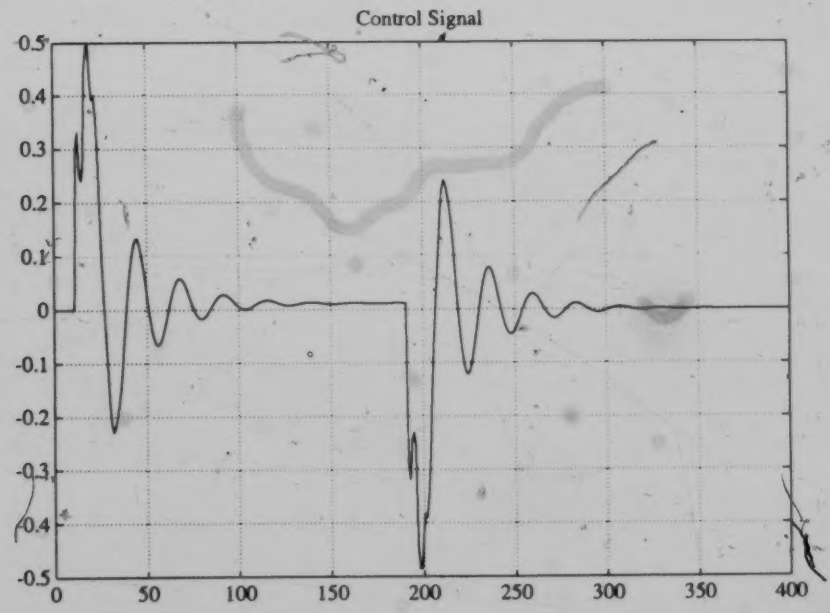


Figure 6.13: Simulation Run 11.

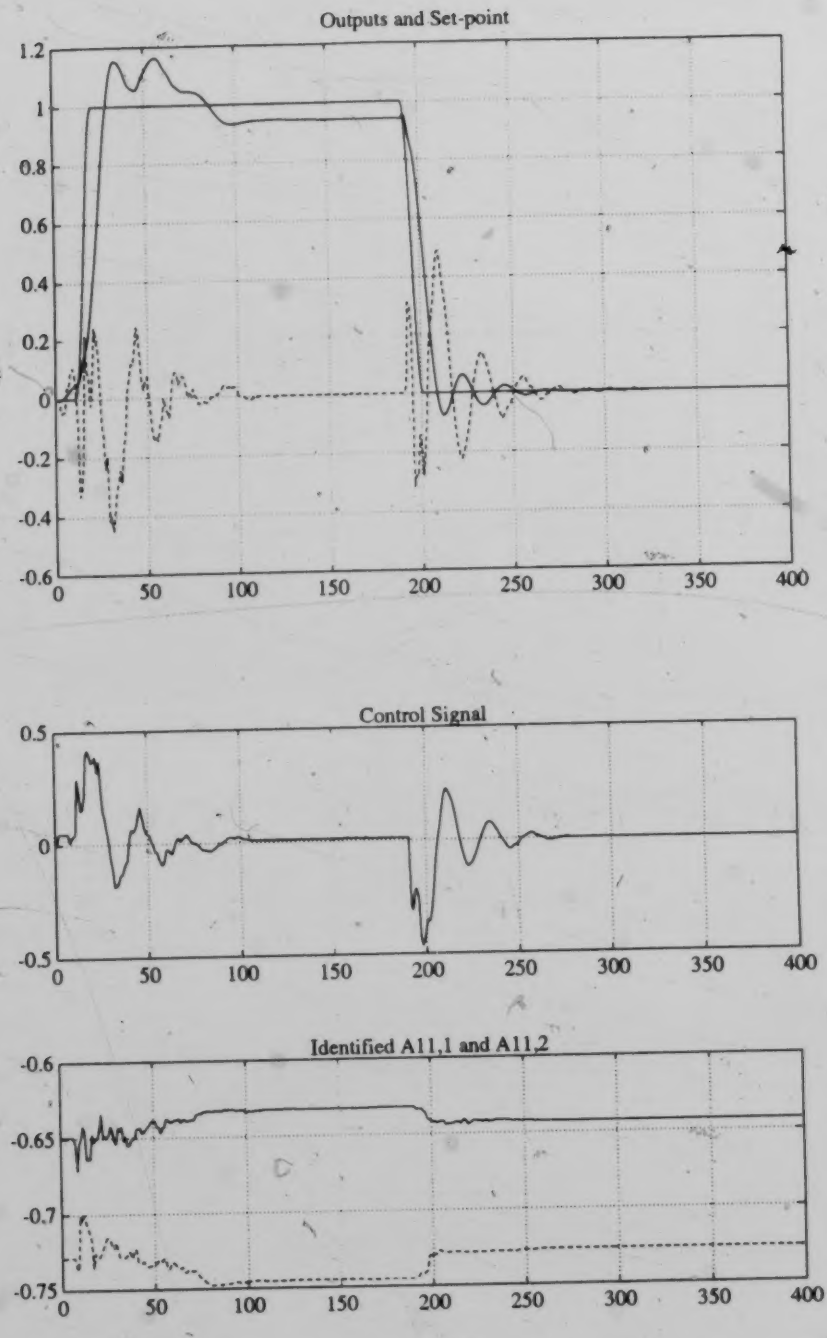


Figure 6.14: Simulation Run 12.

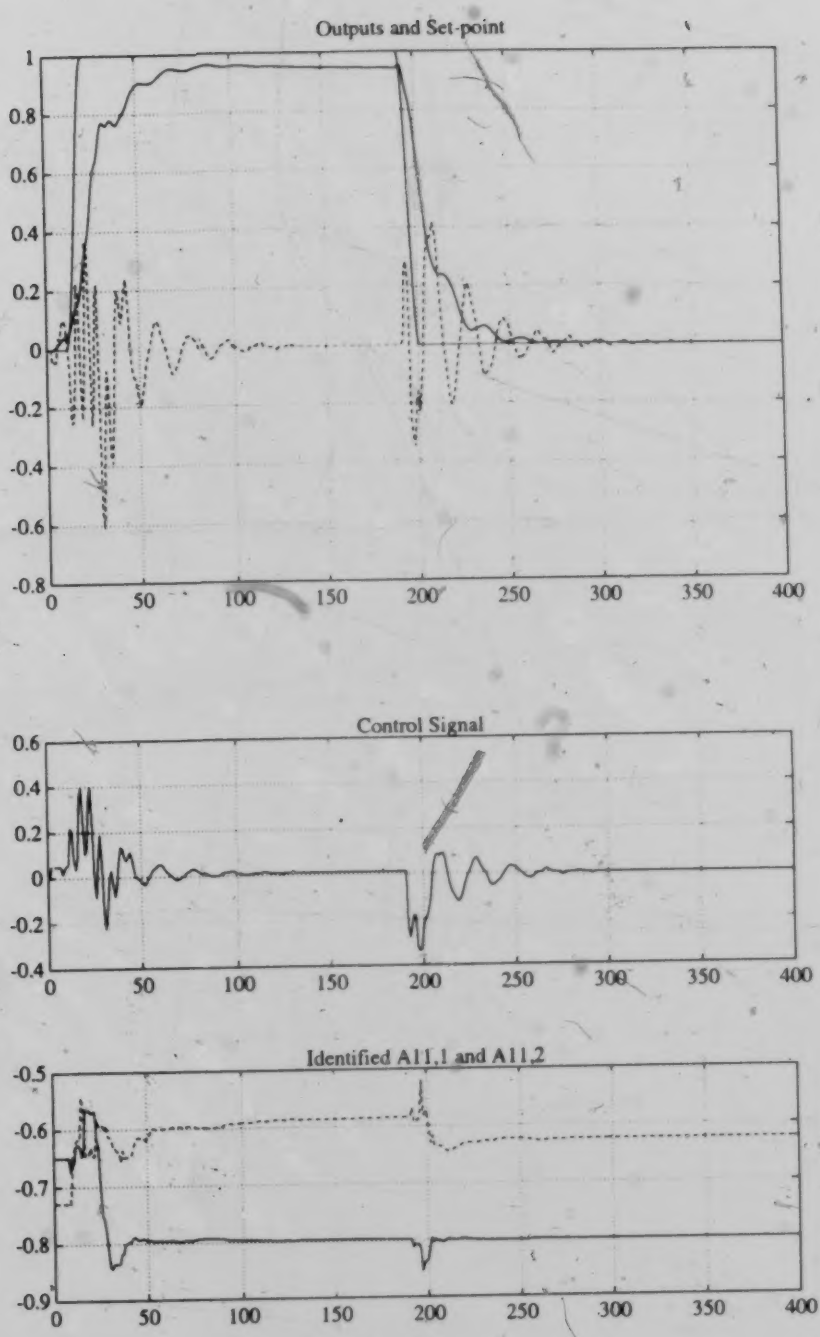


Figure 6.15: Simulation Run 13.

transition when the parameters are already identified. Note that there is no error in the motor position.

Simulation Run 15

When the same tuning knobs were used and both T-filters were removed, the system response became more oscillatory and the error in the motor position was present as shown in Figure 6.17.

Simulation Run 16

The simulation run in Figure 6.18 uses the same settings as the previous simulation run except that $N2_1 = 15$ and $N2_2 = 8$. This resulted in well controlled acceleration but very sluggish motor position response. Note that a significant steady state error in motor position can be observed again.

Simulation Run 17

To improve the dynamics of this response, the control horizon $NU_1 = 4$ was used. Figure 6.19 shows faster but more oscillatory response.

Simulation Run 18

Finally, the simulation run shown in Figure 6.15 is repeated with the intention of removing the motor position steady state error. Therefore, everything is identical except for output error weighting coefficients which were chosen $\lambda_{y1_{11}} = 1$, $\kappa_1 = 1$, $\lambda_{y2_{11}} = 1$, and $\kappa_1 = 1.5$. This, as can be seen in Figure 6.20, removed the steady state error but increased the amplitude of the acceleration.

6.5 Summary

The experience from the simulation runs presented in this chapter can be characterized by the following points:

- The MIMO GPC can control nonsquare systems with $m < n$ if the control requirements are not contradictory.
- General guidelines suggested by Clarke et al. for the SISO GPC [25] for the choice of the tuning knobs are generally valid for the MIMO GPC but must take into account the multivariable character of the system.
- Use of the GPC and RPEM T-filters is beneficial for the performance of the MIMO GPC but the form of the T-filters must respect the actuator and plant dynamics as well as the expected plant/model mismatch, which are difficult to predict.

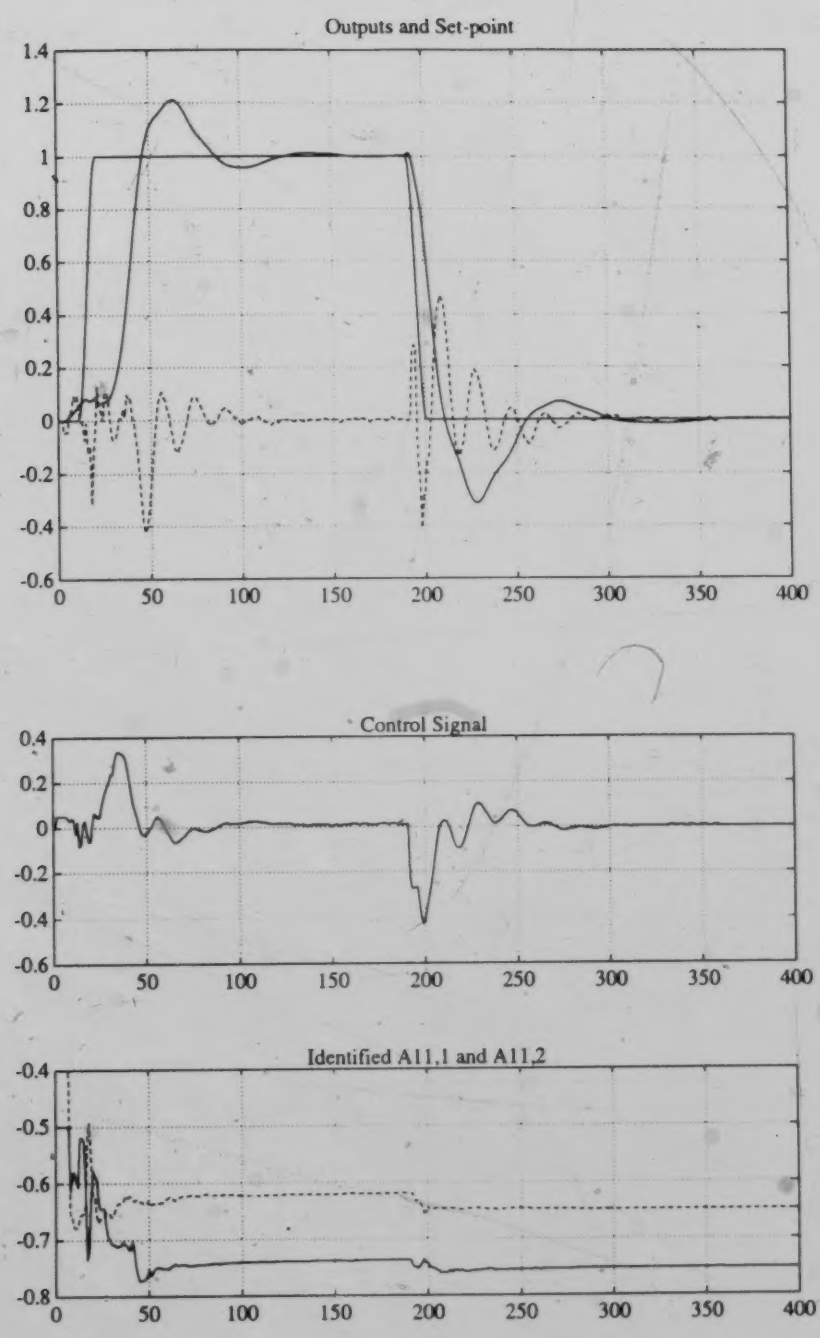


Figure 6.16: Simulation Run 14.

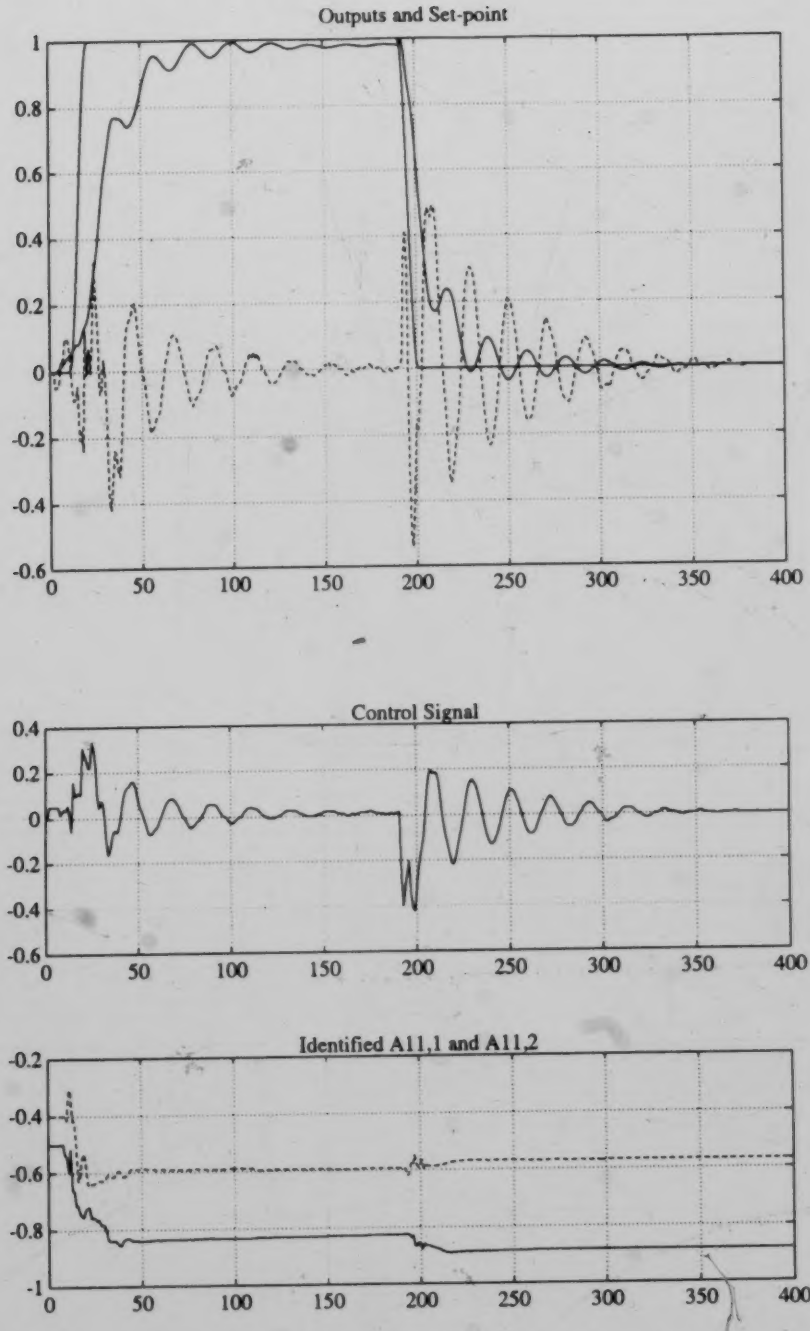


Figure 6.17: Simulation Run 15.

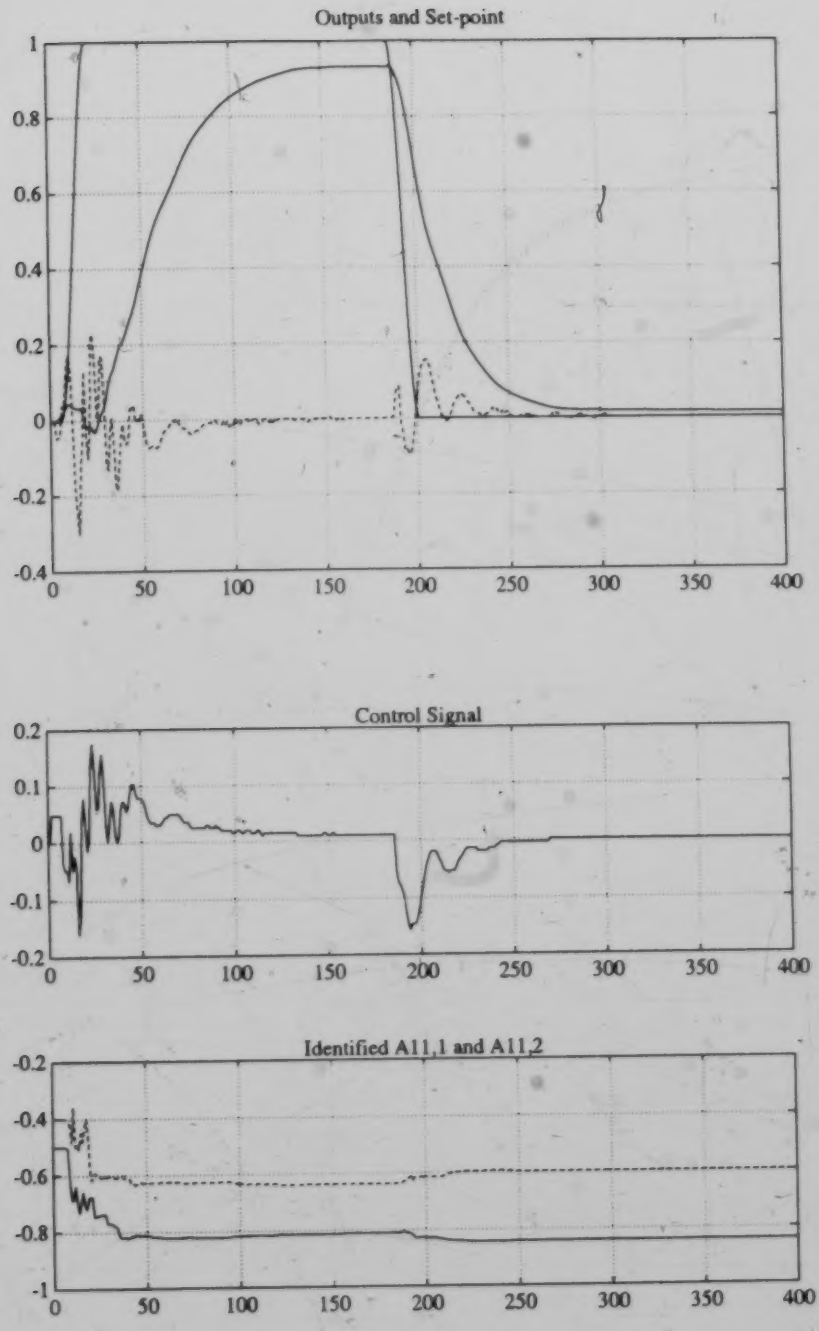


Figure 6.18: Simulation Run 16.

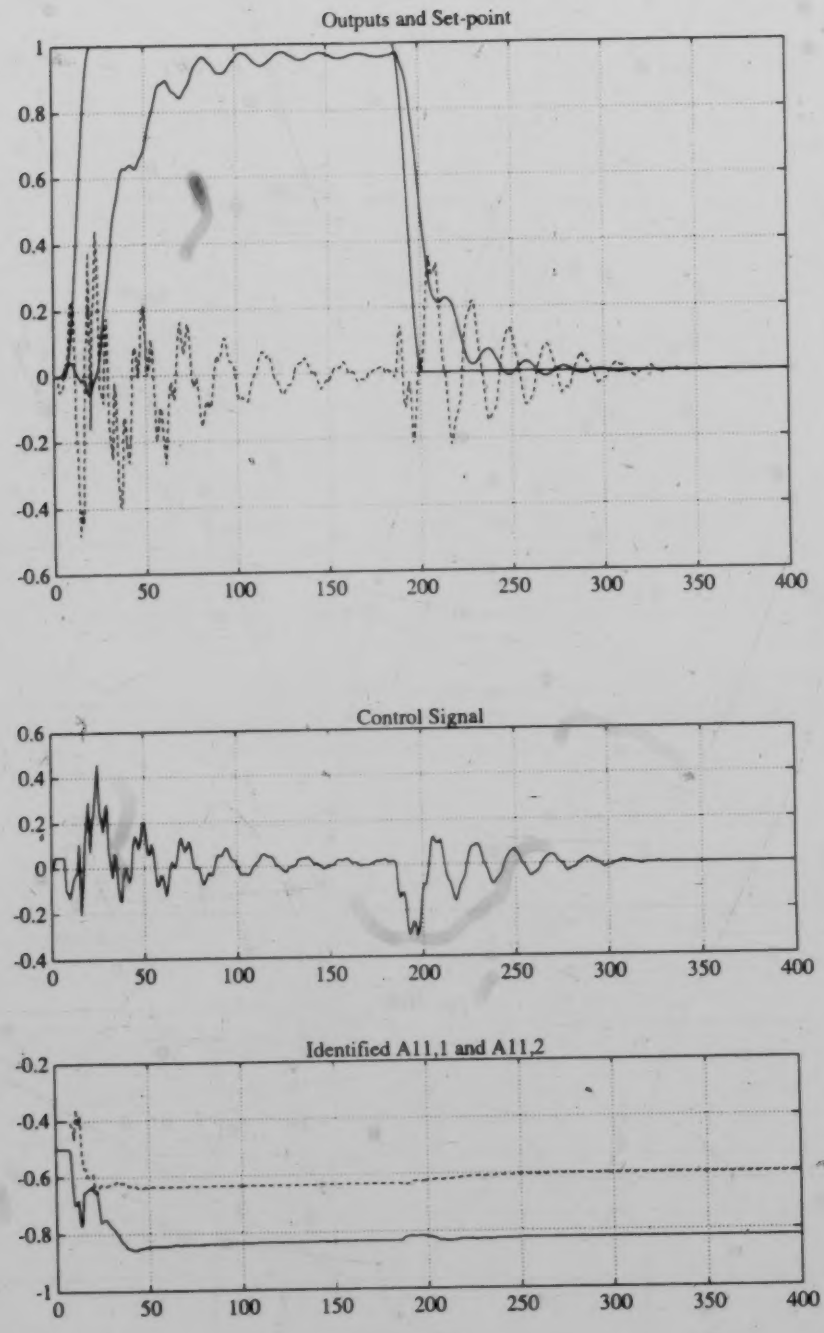


Figure 6.19: Simulation Run 17.

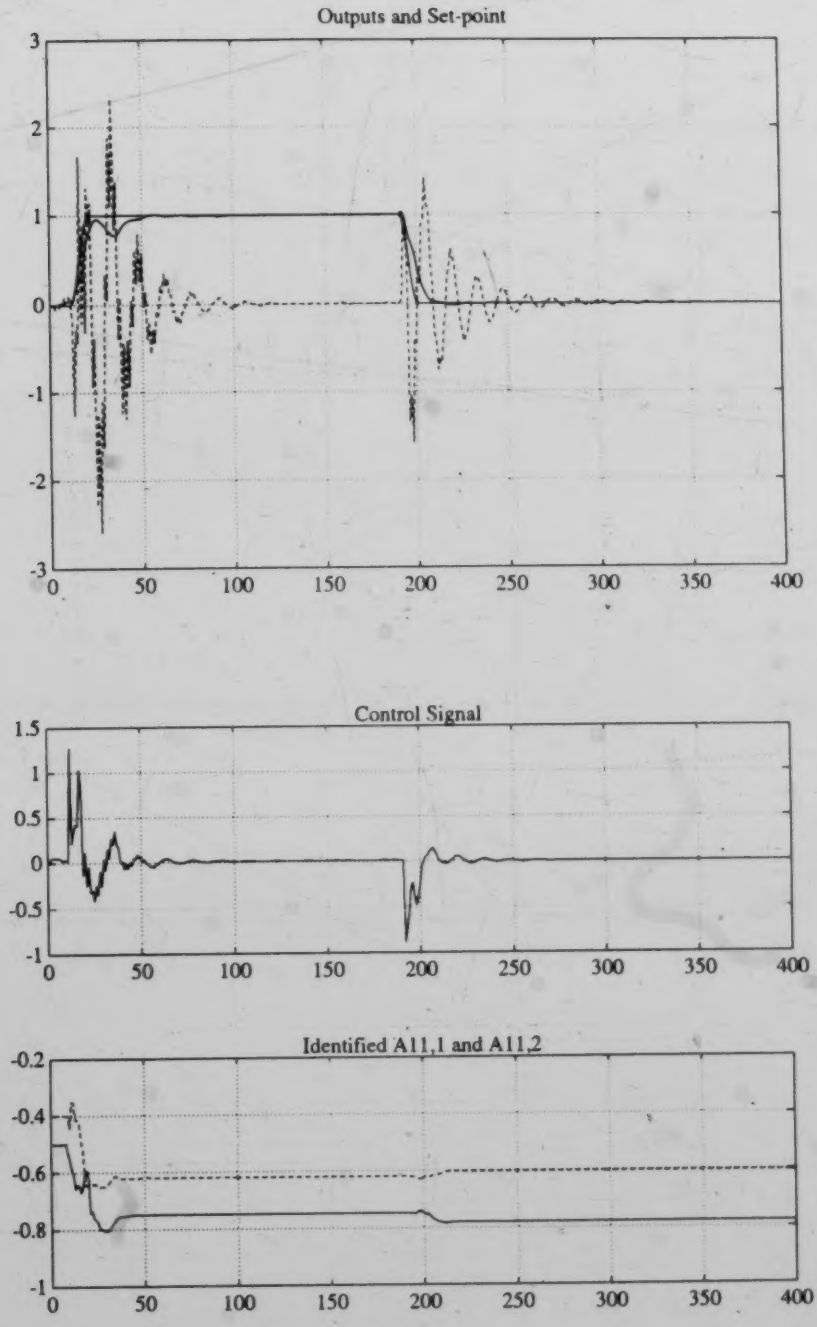


Figure 6.20: Simulation Run 18.

- The output error weighting coefficients have similar beneficial impacts on the closed-loop performance of the MIMO GPC algorithm as the T-filters.
- The MIMO output coupled plants are more difficult to control in case of the plant/model mismatch and the diagonal terms $A_{ii}(q^{-1})$ should be overparameterized and the coupling terms $A_{ij}(q^{-1})$ should be underparameterized.
- The T-filters and the output error weighting coefficients help the MIMO GPC to cope with underparameterized plants.
- The minimum control horizon NU_i can be smaller than the number of badly located poles in case of the nonsquare systems with $m < n$.
- In the case of the MIMO GPC, output errors can be minimized by the output error weighting coefficients. However, any change in the plant/model mismatch may introduce additional output errors which require different values of the output error weighting coefficients.

In conclusion it can be said that under reasonable noise and plant/model mismatch, the MIMO GPC performs well. However, model parameterization and the choice of tuning knobs are more difficult to determine than in the SISO case.

Chapter 7

Experimental Results

7.1 Introduction

Many things could be said about comparison of simulation and experimental runs. Without going into a lengthy discussion, it can be stated that experimental verification is the ultimate test to determine if a chosen control law can cope with the plant and real conditions.

In any experimental research project, there is the likelihood of unexpected behavior caused by errors in control algorithm implementation. In the case of the flexible manipulator, the actuators are capable of large torques in a short time, which is a desirable feature. However the possibility always exists that during the initial stages of implementation, an algorithm may fail and drive the flexible manipulator outside the manipulator's allowed workspace activating the brakes. If the flexible manipulator happens to be moving fast at this time, the sudden braking will result in bending of the flexible link. Consequently, great care had to be exercised when software changes were tried for the first time.

Experimental results for two control laws used for controlling the horizontal motion of the flexible manipulator are presented in this chapter. The results of the application of the adaptive MIMO GPC are described in section 7.2. Section 7.3 presents experimental results with a multivariable PID controller. Conclusions are summarized in section 7.4.

7.2 Adaptive MIMO GPC

The basic difficulty in controlling the position of the tip of the flexible link is that it cannot be directly measured. However, the position of the motor shaft and the flexible link tip acceleration are measurable with currently available sensors. Therefore the control law combines these two measurements to achieve the desired objective, namely the tip position control by controlling the motor shaft position and damping out the tip acceleration. When the motor shaft position approaches the desired position and the tip acceleration is zero, it is concluded that the tip is at the desired final position and does not vibrate. This strategy increases the dimensionality of the controlled plant because two outputs must be controlled simultaneously.

When real experiments using adaptive algorithms are performed, the initial values of the estimated parameters should not be zero. It is sufficient if some of the parameters are initialized with values which are in the neighborhood of the real values. Therefore, in the experiments described below some parameters are initialized by values obtained from previous experiments although their sampling frequencies are different. Note that when dealing with RREM, the regressor and gradient vectors are initialized during the first few k sampling periods when the manipulator is driven by a small control signal in an open-loop configuration. The value of k is determined as

$$k \geq \max\left(\deg(A_{ij}), \deg(B_{ij})\right) \qquad \forall i, j \tag{7.1}$$

These steps are undertaken to avoid the initial transient which can be observed from the simulation examples.

Although the simulation results in chapter 6 demonstrate the ability of the MIMO GPC to control complicated systems, it may be optimistic to expect similarly good results from experimental applications of the MIMO GPC. The simulated plants, although quite complex, were linear while the manipulator dynamics are nonlinear. The simulated plants did not exhibit any delay which is always present in real plants. Plant/model mismatch due to the identified parameters plays, therefore, the most important role in successful application of the adaptive MIMO GPC.

Several experimental runs are presented in this section. For obvious reasons, the reasonably successful ones are described. It must be admitted however that not all runs were successful especially when suitable T-filters and output error weighting coefficients were sought.

Because the sampling frequency of the experimental runs was in the range

$< 40, 50 >$ Hz, the 4-th Bessel filter corner frequency was chosen 5 Hz.

GPC: Experimental Run 1

Figure 7.1 shows the experimental run for the following model structure: deg(A_{11})=4, deg(A_{12})=2, deg(B_{11})=4, deg(A_{21})=2, deg(A_{22})=2, and deg(B_{22})=4. The GPC tuning knobs and sampling frequency are: $N1_1 = N1_2 = 1$, $N2_1 = 10$, $N2_2 = 15$, $NU_1 = 2$, $\lambda_{y1_{11}} = 0.01$, $\kappa_1 = 0.5$, $\lambda_{y2_{11}} = 2.0$, $\kappa_2 = 0.9$, $T_{RPEM}(q^{-1}) = 1 - 0.8q^{-1}$, $T_{GPC}(q^{-1}) = 1 - 0.8q^{-1}$, and $f_s = 40$Hz. The overall performance is reasonable except for the initial overshoot of the motor shaft position which may be attributed to inaccurate initial parameter estimation. The sampling frequency 40 Hz was chosen due to the computational load of the entire control algorithm. Additional explanation is needed for the choice of the output error weighting coefficients. The position setpoint is 0.2 radian and a typically observed position error is in the range $< 0, \pm 0.15 >$ radian. It is not unusual to see the acceleration error in the range $< 0, \pm 5 >$V or more. This means that the ratio of the acceleration error to the shaft position error can be large at certain sampling instants. Although the ratio is not the best measure for the choice of output error weighting coefficients for the MIMO, it can be taken as the initial guideline. It is desirable to specify the motor position as the more important signal because the goal is to reach the final motor shaft position and then to damp out the acceleration while the motor shaft moves around the final position. The control signal in Figure 7.1 is quite active. Some of the identified parameters of the polynomial $A_{11}(q^{-1})$ are shown to emphasize that it is the prediction of the tip acceleration which is the troublesome part for the MIMO GPC. The identified coefficients of the polynomial $A_{22}(q^{-1})$ representing the motor dynamics can be seen in Figure 7.2 and are shown only in this experiment.

GPC: Experimental Run 2

To investigate if a lower order model can be used, the system was parameterized diagonally as follows: deg(A_{11})=2, deg(B_{11})=3, deg(A_{22})=2, and deg(B_{22})=3. The remaining tuning knobs were chosen as : $N1_1 = N1_2 = 1$, $N2_1 = 12$, $N2_2 = 14$, $NU_1 = 2$, $\lambda_{y1_{11}} = 0.01$, $\kappa_1 = 0.75$, $\lambda_{y2_{11}} = 1.0$, $\kappa_2 = 0.9$, $T_{GPC}(q^{-1}) = 1 - 0.8q^{-1}$, and $f_s = 40$Hz. The MIMO RPEM was used in its positional form. The resulting response is shown in Figure 7.3. This 'underparameterization' exhibits more oscillatory behavior and the variance of the individual parameters is large. This response demonstrates how the adaptive MIMO GPC tries to cope with this difficult plant.

GPC: Experimental Run 3

Figure 7.4 shows the response of the flexible manipulator with a different diago-

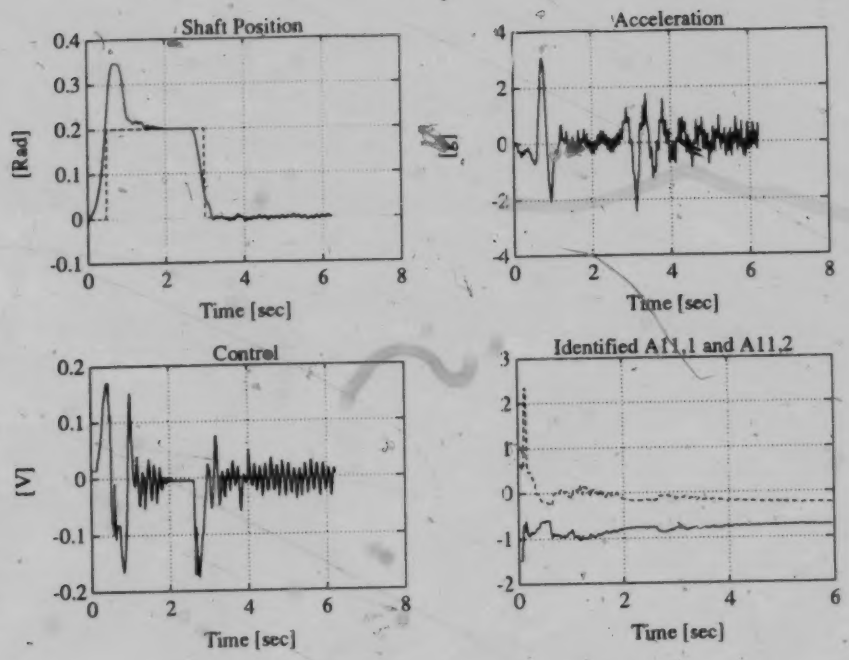


Figure 7.1: GPC: Experimental Run 1.

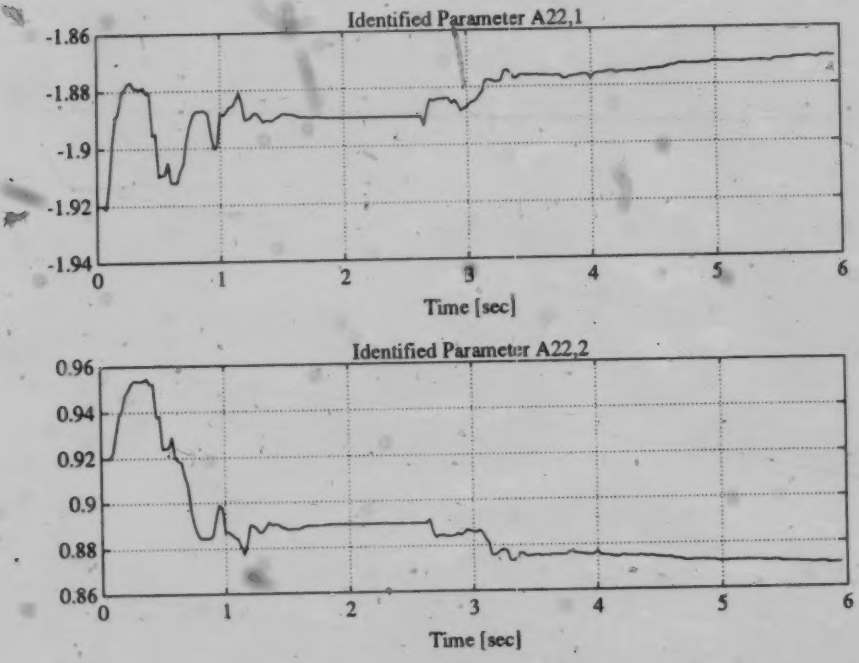


Figure 7.2: Identified $A_{22}(q^{-1})$ polynomial.

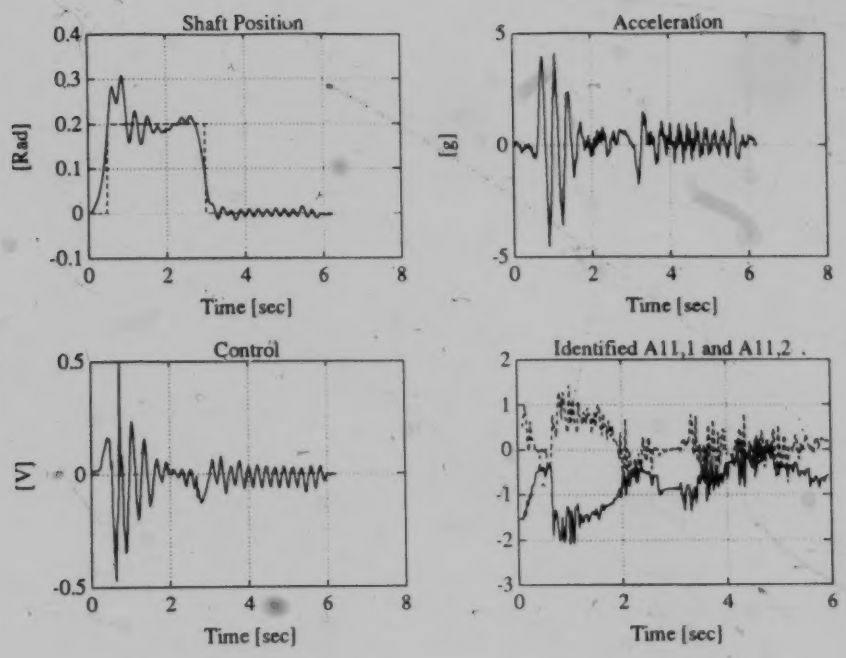


Figure 7.3: GPC: Experimental Run 2.

nalized parameterization: deg(A_{11})=4, deg(B_{11})=4, deg(A_{22})=2, and deg(B_{22})=4. The tuning knobs are : $N1_1 = N1_2 = 1$, $N2_1 = 12$, $N2_2 = 15$, $NU_1 = 3$, $\lambda_{y1_{11}} = 0.01$, $\kappa_1 = 0.5$, $\lambda_{y2_{11}} = 2.0$, $\kappa_2 = 0.9$, $\lambda_u = 0.5$, $T_{RPEM}(q^{-1}) = 1 - 0.8q^{-1}$, $T_{GPC}(q^{-1}) = (1 - 0.05q^{-1})^2$, and $f_s = 40$Hz. The response is very oscillatory and the control signal is aggressive due to $NU_1 = 3$ and due to large control bandwidth defined by the $T_{GPC}(q^{-1})$. When the motor shaft returns to its initial position, the control effort is dominated by the tip acceleration control which, in this case, increases the tip acceleration. In spite of it, the identified parameters are well behaved.

GPC: Experimental Run 4

Figure 7.5 shows an extremely good response of the motor shaft position which demonstrates that the MIMO GPC is a high performance control algorithm. The acceleration amplitude is reasonably small in the last 3 seconds of this experiment but its variance is large, i.e. high frequency vibrations. The diagonal model parameterization for this run is: deg(A_{11})=4, deg(B_{11})=4, deg(A_{22})=2, and deg(B_{22})=3. Tuning knobs: $N1_1 = N1_2 = 1$, $N2_1 = 8$, $N2_2 = 1j$, $NU_1 = 3$, $\lambda_{y1_{11}} = 0.01$, $\kappa_1 = 0.8$, $\lambda_{y2_{11}} = 2.0$, $\kappa_2 = 0.9$, $T_{RPEM}(q^{-1}) = (1 - 0.4q^{-1})^2$, and $T_{GPC}(q^{-1}) = (1 - 0.05q^{-1})^2$.

GPC: Experimental Run 5

The prediction horizon for the tip acceleration is lowered more and Figure 7.6 shows a

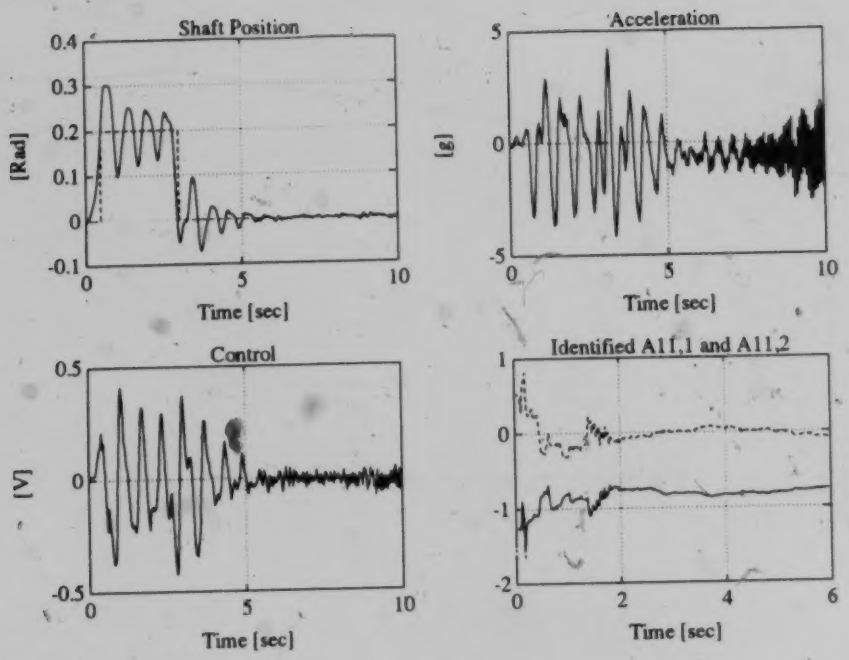


Figure 7.4: GPC: Experimental Run 3.

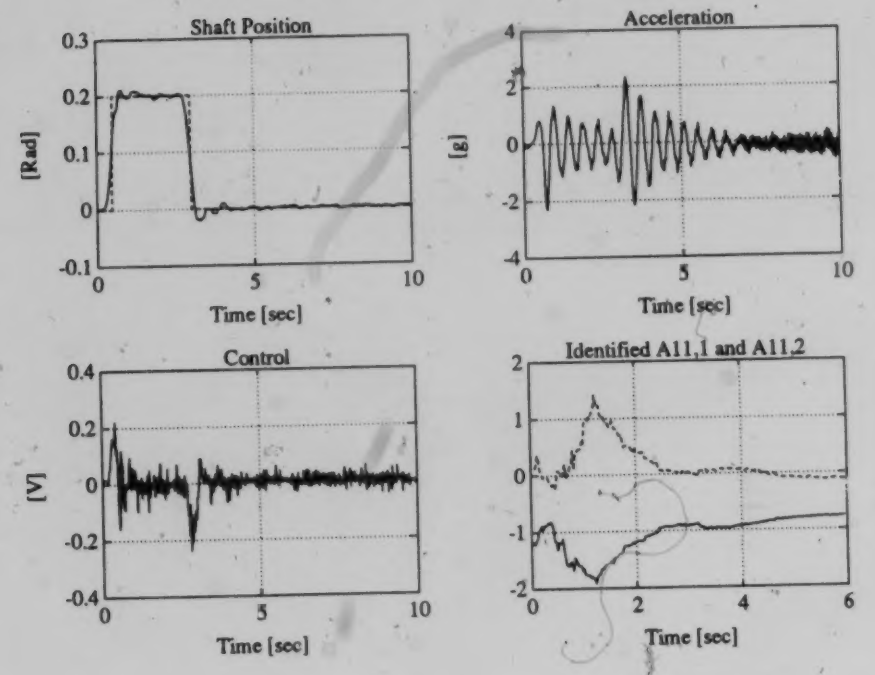


Figure 7.5: GPC: Experimental Run 4.

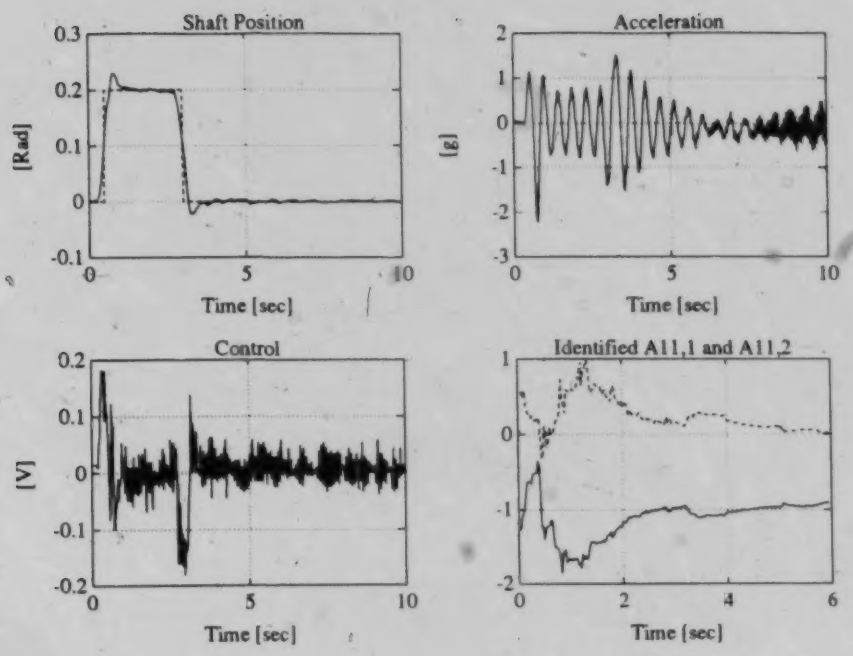


Figure 7.6: GPC: Experimental Run 5.

good response. The motor position performs extremely well and the tip acceleration remains small during the entire run. The intensive control is aimed at reducing the tip acceleration. All parameters and tuning knobs are the same as in the previous run, the only change is $N2_1 = 5$.

GPC: Experimental Run 6

The experimental run in Figure 7.7 was aimed at better identified parameters before a larger transition in motor shaft position was executed. The entire parameterization and tuning knobs are identical to the previous experimental run, only the position trajectory was changed. First, only a small step of 0.05 radian is commanded to provide the identification procedure the opportunity to identify the system parameters. Then, after the motor shaft returns to its initial position, larger motor shaft position step of 0.4 radian was commanded. As it can be seen, the overall response was acceptable.

GPC: Experimental Run 7

The experimental run in Figure 7.8 repeats the same position trajectory. The main changes are $N2_1 = 3$ and $\lambda_{y1_{11}} = 0.1$. The system parameterization is: deg(A_{11})=4, deg(B_{11})=4, deg(A_{22})=2, and deg(B_{22})=3. The tuning knobs are: $N1_1 = N1_2 = 1$, $N2_1 = 3$, $N2_2 = 10$, $NU_1 = 2$, $\lambda_{y1_{11}} = 0.1$, $\kappa_1 = 1.0$, $\lambda_{y2_{11}} = 1.0$, $\kappa_2 = 1.2$,

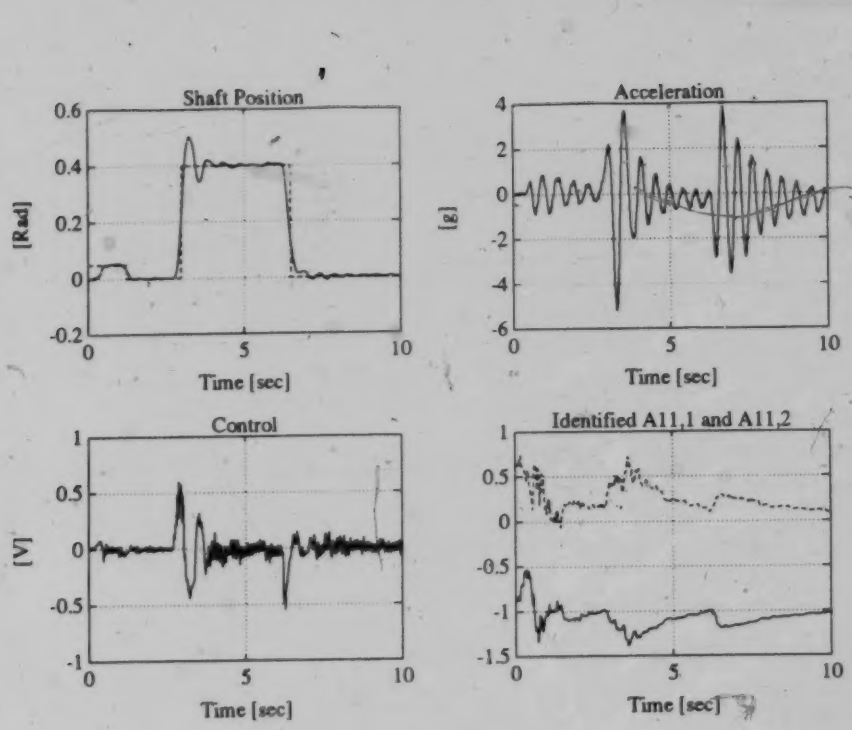


Figure 7.7: GPC: Experimental Run 6.

$T_{RPEM}(q^{-1}) = (1-0.4q^{-1})^2$, and $T_{GPC}(q^{-1}) = (1-0.05q^{-1})^2$. Although the acceleration error weighting coefficient is significantly increased, the short prediction horizon $N2_1$ and $\kappa_2 = 1.2$ are reasonably balancing this increased acceleration error weighting coefficient. The position response is very good and the acceleration is attenuated but not completely.

GPC: Experimental Run 8

Finally, Figure 7.9 shows the experimental run which has the same parameterization and tuning knobs like the previous experiment with the exception that $N2_1 = 1$, $\lambda_{y1_{11}} = 0.15$ and $\kappa_1 = 1.0$. Although the acceleration was kept fairly small, the motor shaft position exhibited much larger overshoot than in the previous case. This experimental run clearly demonstrates that the control of nonsquare systems, even in simulations, must be seen as a compromise.

7.3 Multivariable PID Control Law

The PID control law is the most widely accepted and used control law. To control the horizontal motion of the flexible manipulator, the PID control law has been

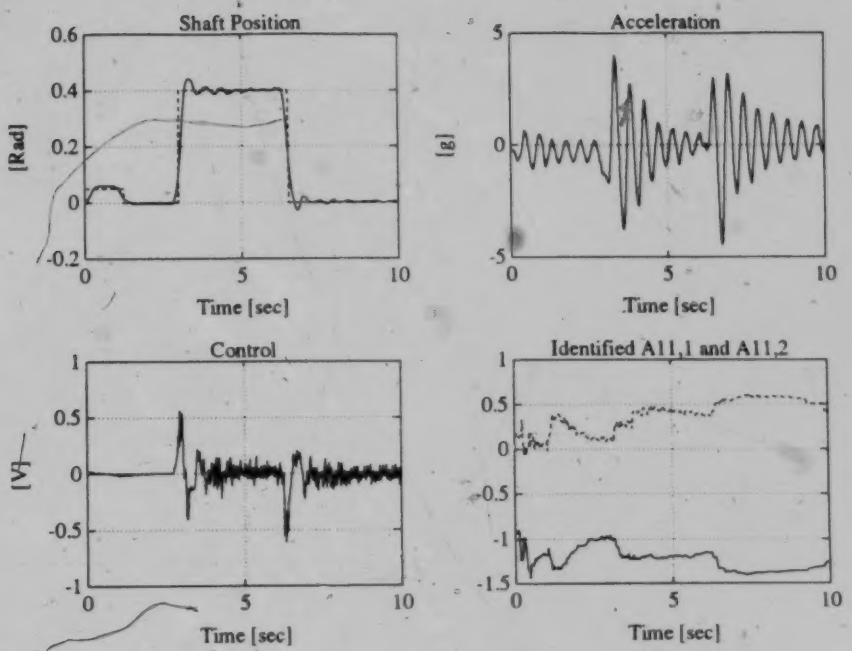


Figure 7.8: GPC: Experimental Run 7.

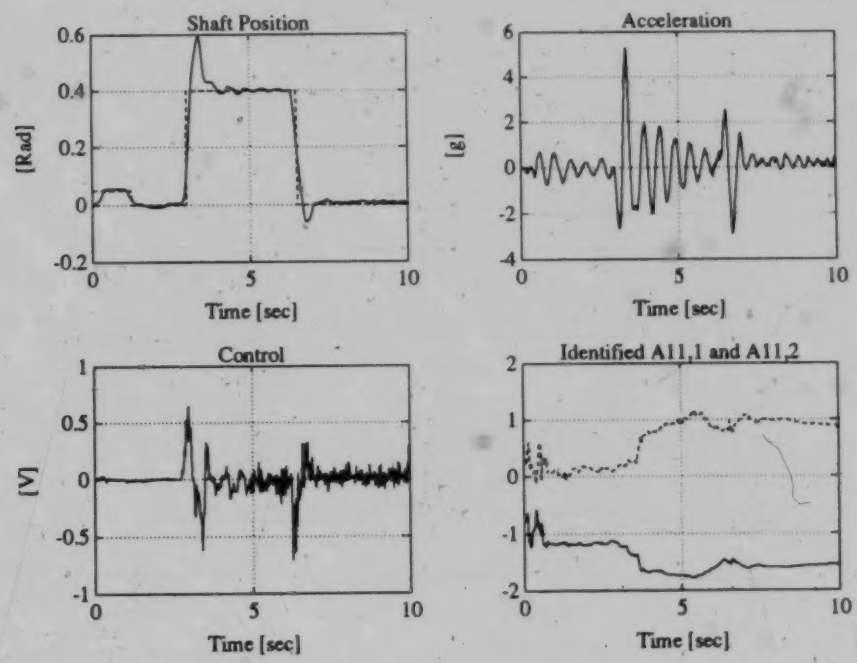


Figure 7.9: GPC: Experimental Run 8.

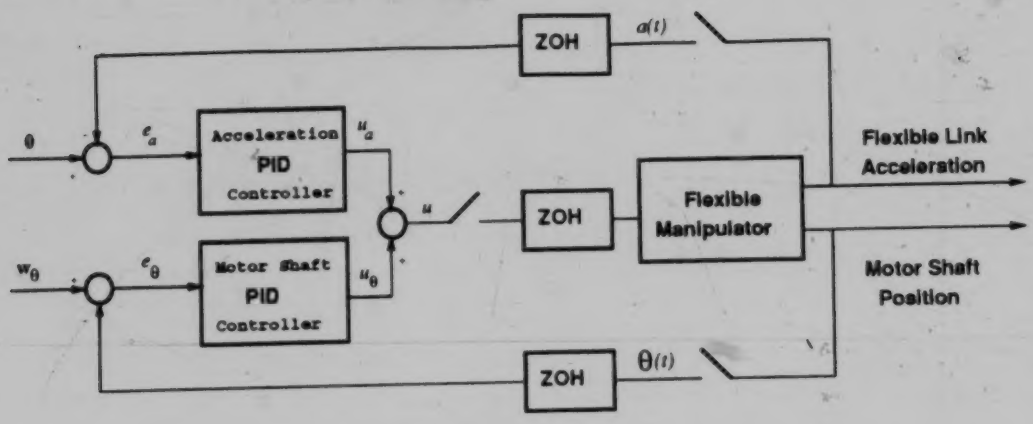


Figure 7.10: Multivariable PID controller - block diagram.

implemented in the multivariable form described by the following set of equations

$$e_\theta(t) = w_\theta(t) - \theta(t) \qquad e_a(t) = 0 - a(t) \tag{7.2}$$

$$u_{P_\theta}(t) = K_{P_\theta} e_\theta(t) \qquad u_{P_a}(t) = K_{P_a} e_a(t) \tag{7.3}$$

$$u_{D_\theta}(t) = \frac{K_{D_\theta}}{h}\left(e_\theta(t) - e_\theta(t-1)\right) \qquad u_{D_a}(t) = \frac{K_{D_a}}{h}\left(e_a(t) - e_a(t-1)\right) \tag{7.4}$$

or the above equation can be replaced by the output-derivative form shown below:

$$u_{D_\theta}(t) = \frac{K_{D_\theta}}{h}\left(\theta(t-1) - \theta(t)\right) \qquad u_{D_a}(t) = \frac{K_{D_a}}{h}\left(a(t-1) - a(t)\right) \tag{7.5}$$

$$u_{I_\theta}(t) = u_{I_\theta}(t-1) + hK_{I_\theta} e_\theta(t) \qquad u_{I_a}(t) = u_{I_a}(t-1) + hK_{I_a} e_a(t) \tag{7.6}$$

and the final control voltage is obtained by the following set of equations

$$u_\theta(t) = u_{P_\theta}(t) + u_{I_\theta}(t) + u_{D_\theta}(t) \qquad u_a(t) = u_{P_a}(t) + u_{I_a}(t) + u_{D_a}(t) \tag{7.7}$$

$$u(t) = u_\theta(t) + u_a(t) \tag{7.8}$$

The structure of the multivariable PID controller is, in a simplified version, shown in Figure 7.10. The subscript P stands for proportional, I for integral, D for derivative, a for acceleration, and θ for motor shaft position. The variable t denotes the current time instant, h is the sampling period, $a(t)$ is the measured acceleration, $\theta(t)$ is the measured motor shaft position, and $e_a(t)$ and $e_\theta(t)$ are the acceleration and motor shaft position errors, respectively. The meaning of remaining PID settings and variables follow standard PID control law.

For simplicity, the friction compensation and the anti-reset wind-up features are not shown in the above equations although they are implemented.

It is usual to expect the PID settings K_{P_θ}, K_{I_θ}, K_{D_θ}, K_{P_a}, K_{I_a}, and K_{D_a} to be non-negative. The motor shaft PID settings $K_{P_\theta} = 0.95$, $K_{I_\theta} = 0.90$, $K_{D_\theta} = 0.1$, are the same as in the experiments in section 4.3 when the friction compensation effects were investigated.

The usual assumption of non-negative PID settings for the acceleration control were found to be not fully valid. During the final phase of the control experiments the flexible link motion is periodic and can be considered as a superposition of sinusoidal motions with different frequencies. Because, usually, only the first two modal frequencies dominate the vibrations, the acceleration measurements can be approximated by

$$a(t) \approx -\sum_{i=1}^{i=2} A_i \Omega_i^2 \sin(\Omega_i t) = -\sum_{i=1}^{i=2} \Omega_i^2 s_i(t) \tag{7.9}$$

where $s_i(t)$ is the tip position due to the appropriate i-th modal frequency, A_i is the amplitude of the sinusoidal motion, and Ω_i is the angular frequency of the i-th modal frequency. The important observation is the minus sign in the above equation. It is obvious that if the tip position error is, for example, positive then the tip acceleration error $e_a(t)$ is negative. Therefore if the acceleration measurements are used directly with a proportional controller K_{P_a} to minimize the tip position error, a positive feedback will result and lead to unstable behavior of the overall system. This reasoning is valid only for the final stage of the control algorithm when it can be assumed that the tip motion amplitude is small and is controlled by the motor shaft motion. To achieve the goal of minimizing the tip acceleration and stabilizing the tip position, it is necessary to use negative setting values for K_{P_a} and K_{D_a}. The integral constant K_{I_a}, on the other hand, must stay positive. When these guidelines are followed, it is possible to achieve good performance of the multivariable PID controller as will be shown later.

PID: Experimental Run 1

Figure 7.11 shows the system response when only the proportional controller for the tip acceleration is used. The controller setting is $K_{P_a} = -0.08$. The shaft position settles at the desired position of 1 radian in about 3 seconds, after which the controller activity is driven mostly by the acceleration measurements. Note that the acceleration exhibits a small constant value which is due to the accelerometer offset, offset of the 4-th order Bessel filter, and due to the vertical vibration. All experimental runs described in this section were performed with a sampling frequency f_s=100 Hz, the

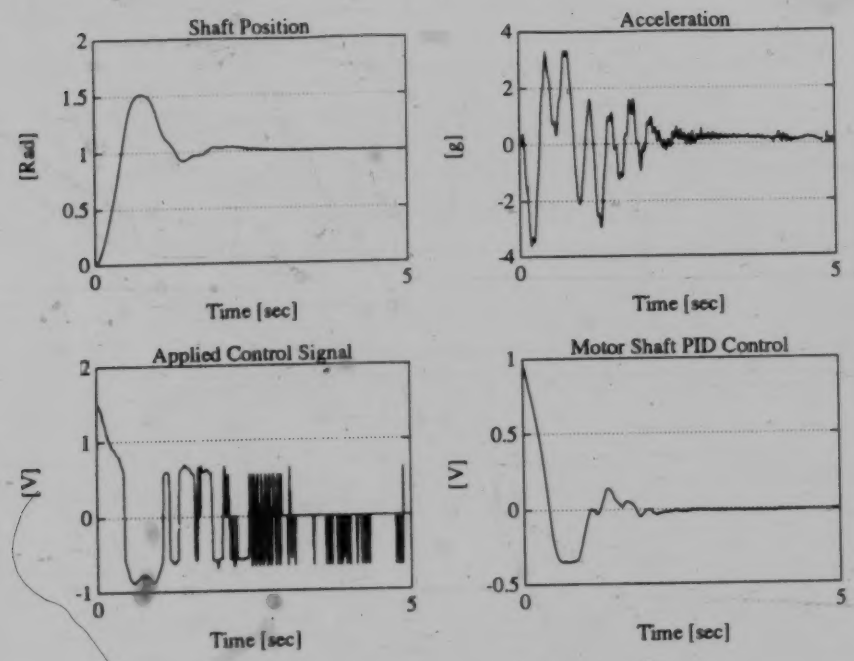


Figure 7.11: PID: Experimental Run 1.

4-th order Bessel filter corner frequency $f_c = 10$ Hz, and command step input of 1 radian. The lower left figure shows the actual applied control voltage which includes the multivariable PID control voltage $u(t)$ together with the friction compensation. The lower right figure shows the control voltage due to the motor shaft PID control without any friction compensation. The setting $K_{P_a} = -0.08$ results in a desirable response with well damped tip acceleration.

PID: Experimental Run 2

If the acceleration P controller setting is $K_{P_a} = -0.18$, the shaft motion as well as the tip acceleration become unstable. The system behavior for $K_{P_a} = -0.18$ is shown in Figure 7.12. Any further decrease of K_{P_a} is not desirable.

PID: Experimental Run 3

Figure 7.13 shows the manipulator response when only the acceleration derivative controller is used with setting $K_{D_a} = -0.04$. The overall behavior is not acceptable and the acceleration is on the verge of instability.

PID: Experimental Run 4

However, when K_{D_a} is changed to $K_{D_a} = -0.01$, the motor shaft position settles at the desired position. Figure 7.14 shows that the control signal is very active which keeps the acceleration quite pronounced. This is not a desirable response and its

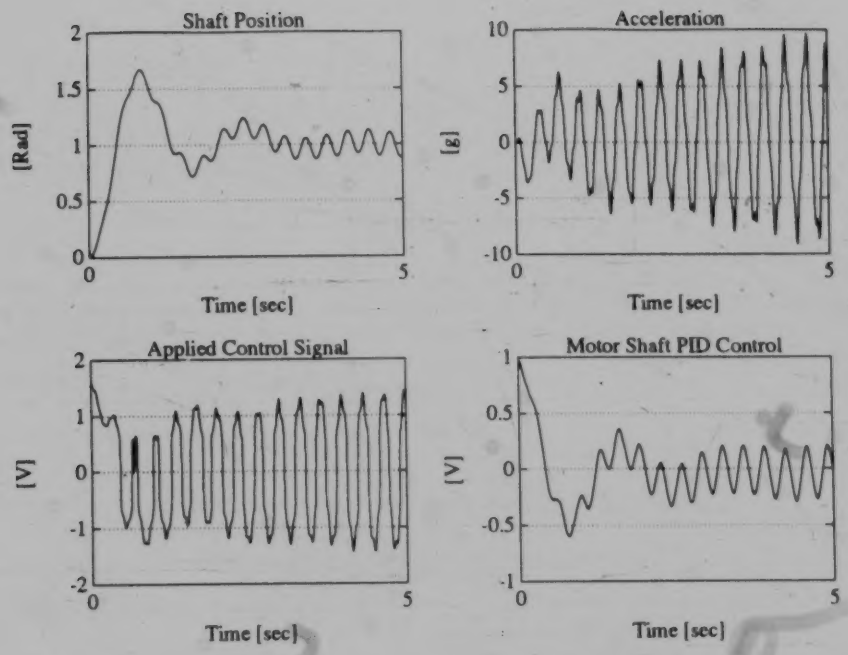


Figure 7.12: PID: Experimental Run 2.

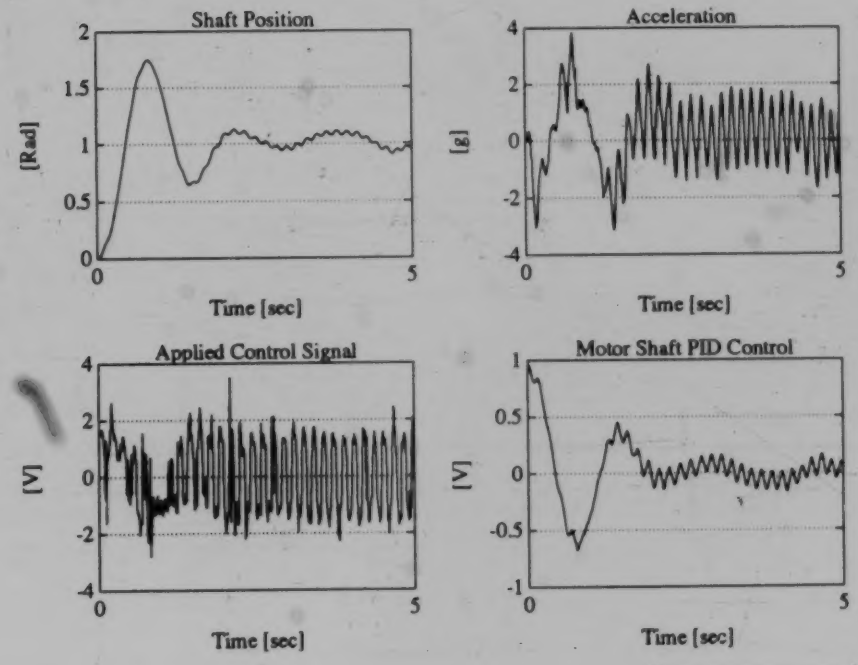


Figure 7.13: PID: Experimental Run 3.

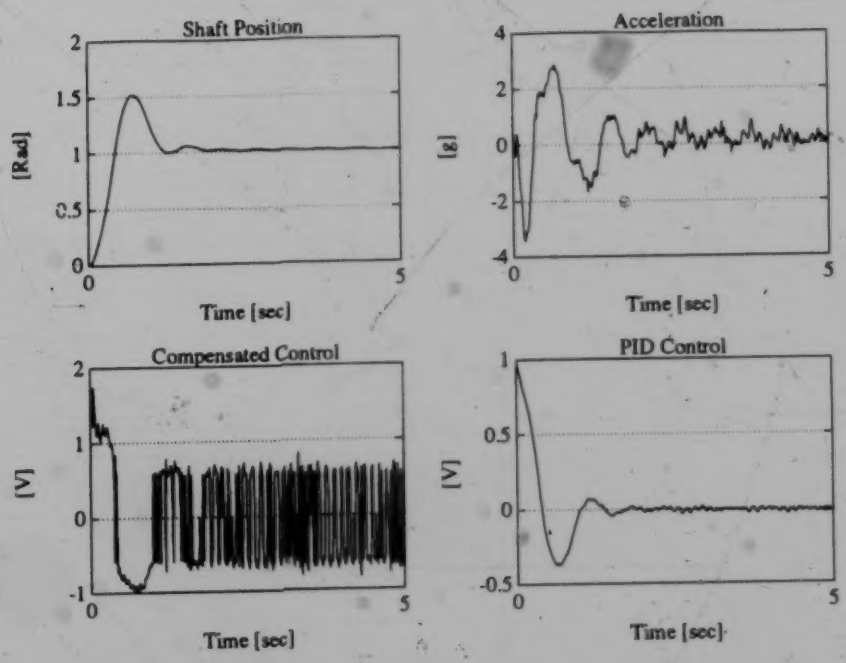


Figure 7.14: PID: Experimental Run 4.

problems are due to the differencing of the acceleration signal which is not smooth and its difference exhibits significant jumps at some sampling instants.

PID: Experimental Run 5

Figure 7.15 shows the experimental run when only the integral acceleration controller was used with settings $K_{I_a} = 0.5$. The result demonstrates problems with the acceleration offset which, if integrated, causes the $u_{I_a}(t)$ to increase with time. This has the consequence that the motor shaft position exhibits a steady state error due to the integrated acceleration offset. The final acceleration is nonzero, but its variation is of a reasonably small amplitude.

PID: Experimental Run 6

The verge of instability, as can be seen in Figure 7.16, occurs for $K_{I_a} = 1.5$. The overall response of the manipulator is totally unacceptable.

PID: Experimental Run 7

Figure 7.17 shows the manipulator response when the accelerometer PID controller is used with settings $K_{P_a} = -0.08$, $K_{I_a} = 0.05$, and $K_{D_a} = -0.01$. The response is excellent if considered within the framework of achieved experimental results.

PID: Experimental Run 8

Equally good response is shown in Figure 7.18 when only the acceleration PD con-

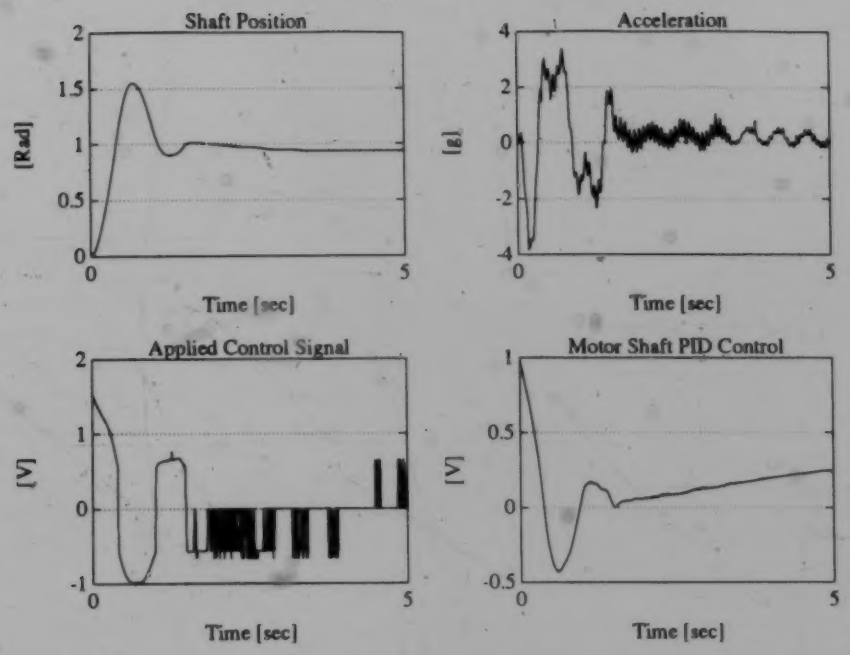


Figure 7.15: PID: Experimental Run 5.

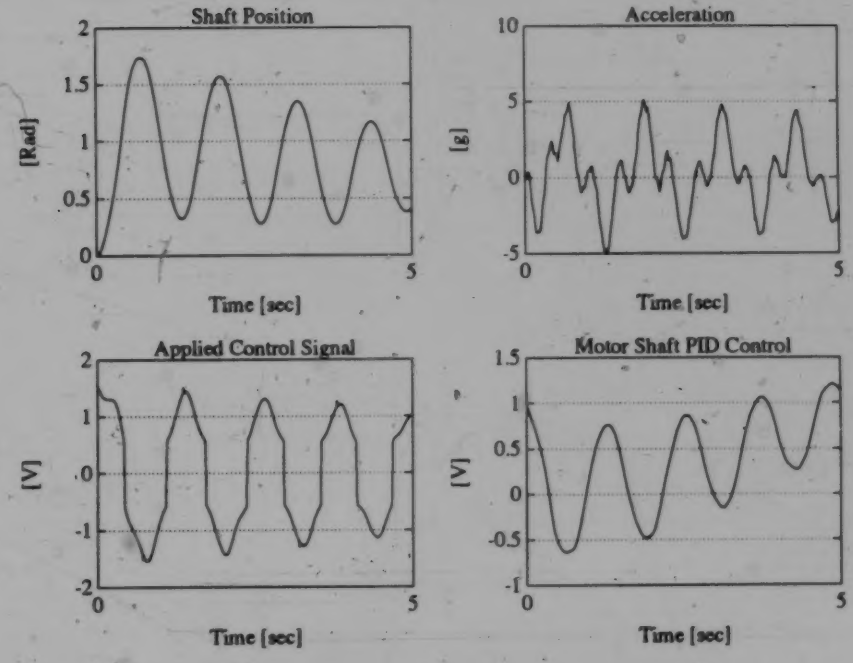


Figure 7.16: PID: Experimental Run 6.

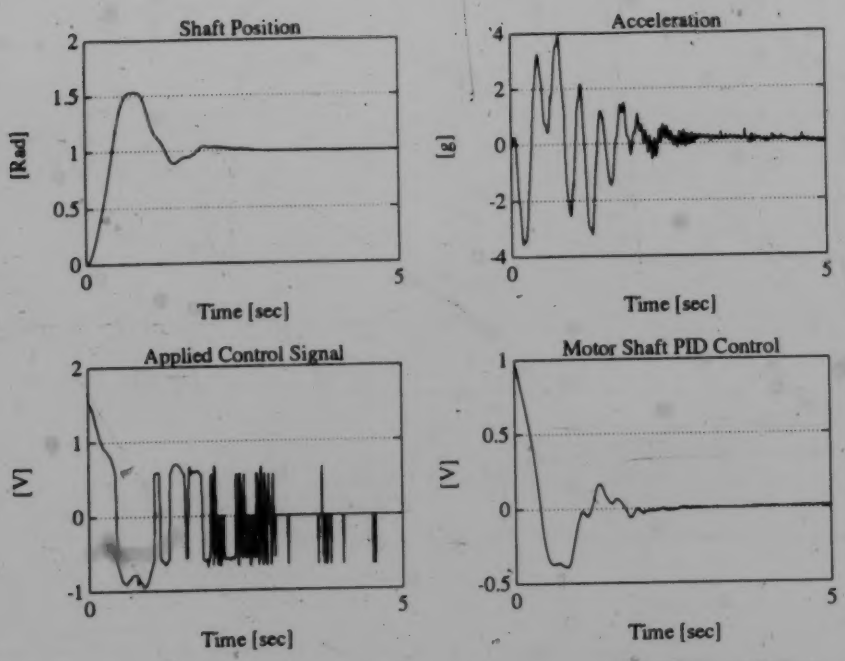


Figure 7.17: PID: Experimental Run 7.

troller is used with settings $K_{P_a} = -0.08$ and $K_{D_a} = -0.01$. It is preferable to use the PD controller so that the effects of the acceleration offset are minimized.

The multivariable PID controller performs well with proper settings considering the quality of the acceleration measurements. The motor shaft overshoot was not minimized because the main goal was to demonstrate the possibility of moving the motor shaft to the desired position and to damp out the tip acceleration. This goal was achieved. The remaining acceleration values are mostly due to the induced vertical vibrations.

In the preceding experiment, the emphasis was on the determination of the acceleration PID controller settings. With the optimum acceleration controller settings thus obtained, one has the freedom to tune the PID controller settings for the motor shaft position, if necessary.

7.4 Summary

The experimental results in this chapter clearly demonstrate that simulation runs are not sufficient to conclude that a given algorithm can control a particular plant. There always will be a need for experimental studies to complement the theoretical

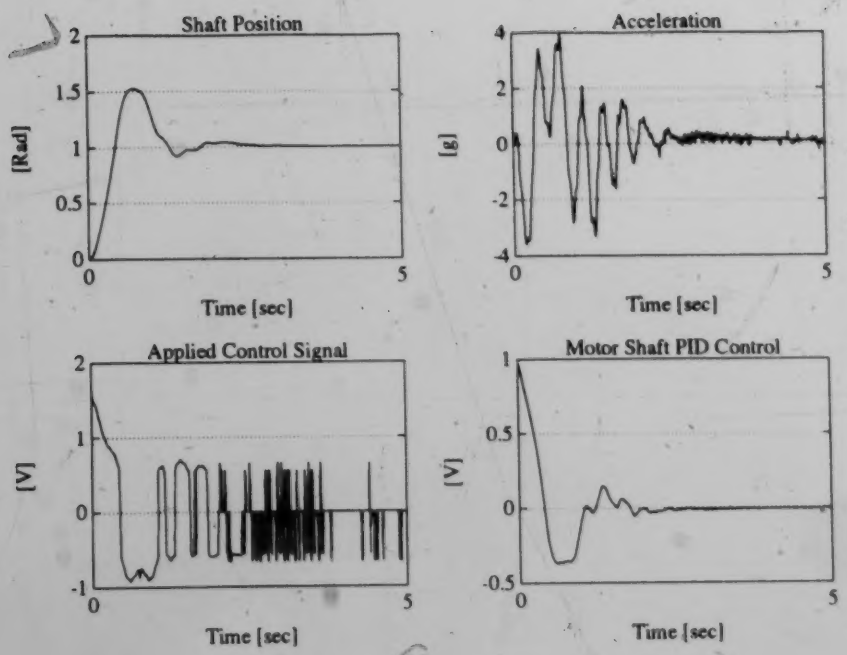


Figure 7.18: PID: Experimental Run 8.

and simulation conclusions.

The experimental runs presented in this chapter did not result in perfect attenuation of the flexible link tip acceleration. However, a properly tuned GPC algorithm demonstrated excellent control of the motor shaft position while the multivariable PID control law attenuated better the acceleration of the flexible link.

The following conclusions and observations can be drawn from the experiments with the multivariable PID controller:

- The integral mode of the acceleration PID controller introduces an error in the motor shaft position due to the acceleration offset.
- Larger settings of the derivative mode of the acceleration PID controller cause more active control signal due to random rapid changes in the acceleration signal.
- The acceleration proportional controller yields the best attenuation of the tip acceleration of the flexible link.
- To achieve a good and stable response, the proportional and derivative mode settings must be negative and the integral settings should be positive.

The experience from the experimental runs of the adaptive MIMO GPC controlling the flexible manipulator can be summarized as follows:

- Stable responses can be only obtained if proper output error weighting coefficients are chosen respecting approximately the ratio of errors in tip acceleration and motor shaft position.
- Use of the incremental version of the MIMO RPEM is preferable to the positional version because it results in a better locally linearized model.
- Lower order parameterization is possible if the T-filter and/or the output error weighting coefficients are used.
- The plant/model mismatch and the measurement noise are the main obstacles in fully successful experimental results of the adaptive MIMO GPC.
- Reasonable responses can be obtained even when output coupled systems are parameterized as output decoupled systems.
- The plant identification plays a crucial role in achieving a successful experimental run.
- Reasonable responses can result from different combinations of the T-filters, output error weighting coefficients, control horizons, and output horizons because their effects are partially overlapping. The choice of a working combination is sometimes obtained by trial and error.

Although perfect damping out of the tip accelerations have not been achieved, it has been attenuated substantially. This can be explained as follows. First, as already mentioned, the friction compensation has an impulse character which in sequence excites the flexible link modal frequencies. Second, the flexible link vibrates not only in the horizontal plane but also in the vertical plane which is sensed by the accelerometer (although quite attenuated) and taken by the MIMO GPC as vibrations in the horizontal plane. Third, the signal coming from the accelerometer signal conditioning unit exhibits variable offset which is difficult to compensate and has profound influence when the acceleration of the flexible tip approaches small values. Fourth, the designed 4-th order Bessel filters introduce noise which becomes relevant when the accelerometer signal voltage amplitude is within range ± 100mV which corresponds to $\pm 0.2g$, i.e. during the final phase of damping out the vibrations.

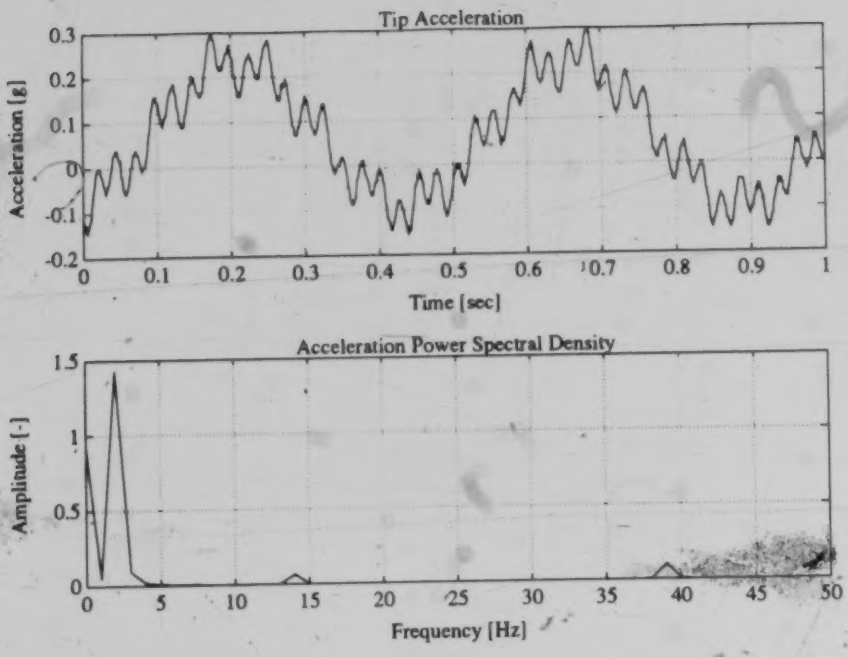


Figure 7.19: Acceleration and its power spectral density when the flexible link vibrates vertically but measurements are in the horizontal direction.

Additional difficulty is presented by the accelerometer transversal cross-talk and the fact that during horizontal vibrations vertical vibrations also exist. Figure 7.19 shows measured acceleration and its frequency spectrum of the flexible link vibrating vertically. As expected the same modal frequencies exist due the the symmetry of the cross-section of the flexible link. The transversal cross-talk represents a real challenge to a successful control algorithm because it is not possible to prevent the existence of the vertical vibrations when a square cross-section is used.

Chapter 8

Parallel Processing and Some Software Implementation Issues

8.1 Introduction

Any successful scheme of real-time implementation of MIMO control algorithms must have the flexibility of achieving different sampling periods. It is necessary to have appropriate hardware and software to determine the time requirements of the various software modules in the MIMO algorithm, in order to assess where the algorithm spends most of its time. This becomes even more important when the order of the system, output and control horizons, as well as other tuning knobs are frequently changed.

It often happens that the entire algorithm cannot be executed on a particular processor within the desired sampling period. Consequently one has to resort to either a very powerful single CPU or to a parallel processing environment. In this chapter parallel processing via a transputer network is described. The discussion also includes some software implementation issues.

This chapter is organized as follows. Section 8.2 discusses timing considerations for implementation of control algorithms. The choice of a transputer network based parallel processing environment is explained in section 8.3. Hardware configuration of the transputer network and the transputer programming environment used in this project are described in section 8.4. Section 8.5 provides details about the developed communication protocol which is used to fully utilize the parallel processing environment. Parallelization of the adaptive MIMO GPC algorithm is explained in section 8.6. Section 8.7 focuses on software debugging considerations in the parallel

processing environment. Considerations for additional speed improvements of the MIMO GPC are explained in section 8.8. Finally, the software development phases of the adaptive MIMO GPC are described in section 8.9.

8.2 Timing Requirements

The Scientific Solution LabMaster PC board, located in the host 25 MHz 80386 based PC, contains among others the 9513A programmable counter-timer. When the MIMO GPC is running, the programmable counter-timer is used to generate hardware interrupts at the desired sampling frequency. When, on the other hand, the timing performance of the algorithm or modules is tested, the 9513A counter-timer is programmed as a hardware counter of input pulses with frequency 1 MHz. Two routines, start and stop counter, are used for measuring the time needed for the execution of a particular section of the algorithm.

When the implementation of the adaptive MIMO GPC is considered, it is expected that significant computational power will be needed for the following sections of the algorithm: calculation of the free responses $\mathbf{p}_i$, calculations of step coefficients, $\sum \mathbf{G}_i^T \Lambda_{yi} \mathbf{G}_i$, $\sum \mathbf{G}_i^T (\mathbf{w}_i - \mathbf{p}_i)$, matrix inversion, and the RPEM identification. However, without actual timing measurements it is difficult to estimate how much time is spent in a particular section of the MIMO GPC algorithm. The timing requirements are important to determine the maximum achievable sampling frequency for a given system parameterization and choice of tuning knobs.

One limitation faced by the author was that the 80386 host computer could not be completely dedicated only to the adaptive MIMO GPC algorithm because it additionally had to look after interrupt handling, software control of all input and output signals, user communication, etc. The initial timing measurements of simulations in one plane indicated that it was possible to achieve a sampling frequency in the range $< 20, 30 >$Hz with reasonable GPC parameter settings. To increase the sampling frequency, it was necessary to consider a more powerful computer with the restriction that it must accept the I/O boards already developed to interface with the flexible manipulator.

The choice of the computational engine is driven by several contradicting requirements. The most important are: price, computational power, programming environment, single processor or multiprocessor, and interface provision for the PC I/O interface boards. In general, the desirable high performance single processor solution tends to be very costly because expensive high speed technologies are used. Less

expensive multiprocessor solutions using standard technologies are associated with complicated software development due to the parallelization of sequential algorithms for MIMD (Multiple Instruction Multiple Data stream) computers.

The MIPS and MFLOPS as well as Dhrystone and Whetstone benchmarks do not provide sufficient information about either the processor performance or timing requirements for a specific application. There are several factors which strongly influence the time needed for execution of a particular algorithm. First, it is the raw performance (MIPS) and architecture of the processor used. Second, the way the algorithm is coded when only sequential processing is considered for only one processor. Third, the overhead due to quality of the programming language used. Fourth, the operating system used. While the above points are valid for both single processor and multiprocessor environments, in the case of the latter two additional factors have to be taken into account. First, the quality of the parallelization of the algorithm when multiple processors are employed which is expressed in the form of a ratio of run time and communication time. Second, the time for synchronization of processes allocated on different processors. It should be expected that variations in the execution time from one system to another will be more pronounced in the multiprocessor systems because the number of variable factors is larger.

All the above mentioned points lead to an obvious conclusion that the correct algorithm timing can be obtained only from measurements made on the actual computer system. Since it is not usually possible, the best performance/cost ratio is used as the main guideline.

8.3 Choice of the Parallel Processing Environment - Transputer

The optimum compromise between price and performance was the MicroWay Quadputer PC board with four 25 MHz T800 transputers (TRANsistor comPUTER) each with 1 MByte memory. Although the transputer features are described in INMOS technical reference [47], some of its specific features are mentioned here. The family of transputers was designed with the goal of having a microprocessor for real-time embedded applications. The original programming language OCCAM as well as the hardware architecture were designed together to follow Hoare's CSP (Communicating Sequential Processes) philosophy [44] for parallel processing. The transputer, therefore, has a microcoded multitasking kernel, i.e. operating system in silicon. The CPU

Processor	80386 & 80387	T800	T800
Clock	25 MHz	25 MHz	25 MHz
Precision	double	double	single
NU=1, N2=10	13677 μsec	9678 μsec	8750 μsec
NU=2, N2=10	17024 μsec	10883 μsec	9763 μsec
NU=3, N2=10	22239 μsec	13301 μsec	11856 μsec
NU=4, N2=10	28905 μsec	16040 μsec	14180 μsec
NU=1, N2=15	19990 μsec	14220 μsec	12852 μsec
NU=2, N2=15	24294 μsec	15734 μsec	14127 μsec
NU=3, N2=15	31117 μsec	18605 μsec	16561 μsec
NU=4, N2=15	39334 μsec	21913 μsec	19344 μsec

Table 8.1: MIMO GPC timing.

Processor	80386 & 80387	T800	T800
Clock	25 MHz	25 MHz	25 MHz
Precision	double	double	single
RPEM	8281 μsec	5073 μsec	4138 μsec

Table 8.2: MIMO RPEM timing.

of the transputer is RISC (Reduced Instruction Set Computer) architecture and the chip also contains the floating point coprocessor. While the traditional approach to multiprocessor systems requires some type of a common bus which is accessed by all processors, the transputer architecture has four high-speed input and output DMA driven serial links which are used for communication with other transputers or external devices. The overall hardware design is such that that under ideal conditions ten subsystems can work fully in parallel, i.e. each input and output link is controlled by its own hardware controller and the CPU and floating point coprocessor can work concurrently. However, the last parallelism, i.e. CPU and floating point coprocessor, can be fully utilized only under control of a low level programming language.

The main benefits in using transputers are seen in the easy expandability of any transputer network and the opportunity to investigate effectiveness of various configurations for a particular algorithm.

The evaluation of transputer performance on some non real-time applications was presented, for example, by Stiles [92] and INMOS [48].

The transputer provides the option to measure time with microsecond resolution. The timing of several simulations of the MIMO GPC and MIMO RPEM is shown in Table 8.1 and Table 8.2.

The absolute measurements are not so important as the relative measurements

because the variable settings of the algorithm can change while the relative performance remains unchanged. Both tables show that a single transputer is faster than the combination of 25 MHz 80386 and 80387 chips when calculations are carried out in double precision format. Note also that the ratio is not a linear function of the input and output horizons and, therefore, a prediction of timing for particular settings can be only approximately interpolated. The last column of both tables depicts the timings when the transputer uses only the single precision format for real numbers. Surprisingly, the difference in timing between single and double precision is relatively small. When the accuracy resulting from single and double precision was compared, there was hardly any difference in all simulation runs. Hence, the single precision is used currently to obtain maximum performance from transputers, although it has to be remembered that a situation may occur when this could lead to numerical problems.

It should be noted that the timings were obtained from simulation runs. If the same settings were used in real experiments, the timing would be larger due to synchronization and communication timings. If more transputers were used, the overall timing would be influenced by the efficiency of parallelization and additional synchronization and communication timings.

8.4 Configuration and Software Support

The configuration of the parallel processing environment is shown in Figure 8.1. The transputer links are connected so that each transputer can communicate directly with remaining transputers. However, there is only one connection between the host 386 computer and the transputer system. This connection is a communication bottleneck because its speed is low due to inefficient hardware on the Quadputer board which converts the serial information into 8 bit parallel information needed for the I/O bus. Although any link can communicate with rate 20 Mbit/sec which, if timing for acknowledgement is included, could result in about 1.75 MByte/sec transfer rate, the maximum rate measured under optimal conditions (program written in assembler on the PC side) is about 250 kByte/sec . However, more realistic measurements showed a performance of about 100 kByte/sec.

Another conclusion from Figure 8.1 is related to the hardware interfacing of transputers with the external world. Barlow et al. [12] discuss three options. First, the interface boards can be located in the host computer I/O space which results in the least expensive and the slowest alternative. Second, the interface can be connected

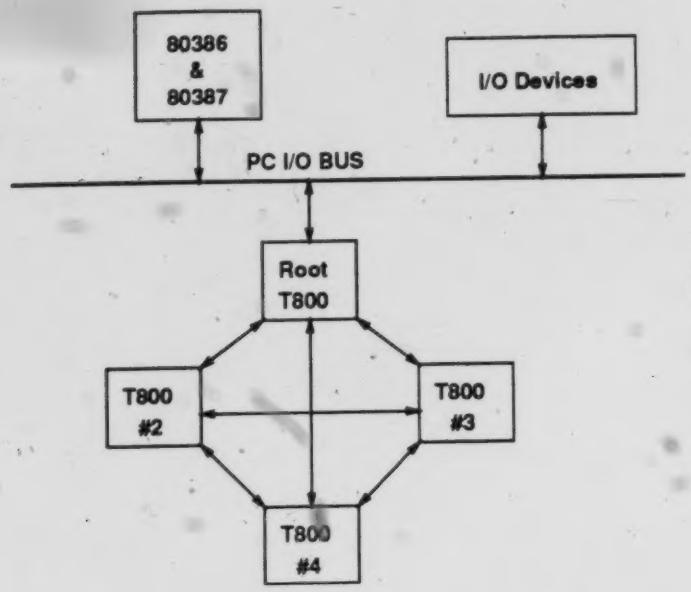


Figure 8.1: Parallel processing hardware configuration.

to a transputer via the transputer serial links. Third, the I/O device interface can be memory mapped in the transputer memory space which is the fastest and the most labor demanding solution because no such products are commercially available. The best solution to parallel processing in controls is reported by Bühler et al. [14] and Whitcomb and Koditschek [106] which consists of several in-house developed transputer based boards in order to standardize interfacing and enable easy modification of the configuration of the transputer network. Since the 2 DOF manipulator interface was originally designed for PC based systems and the transputers as a solution to the demand for higher computational power came later, our interface boards are located on the PC I/O bus. The speed limitation can be overcome to an acceptable degree if the host PC computer controls the I/O boards and does some signal preprocessing so that minimum communication occurs between the host PC computer and the root transputer.

To utilize the transputer parallel system, operating systems or real-time executives are typically used. Their main goal is to simplify the use of the computer system so that a user does not have to deal with many elementary problems to run the computer system and instead can turn his attention to the problems to be solved by the computer system. Currently, there are several transputer operating systems available on the market but they demonstrate similar shortcomings when they are used in systems similar to ours. First, they are mostly intended for computationally

heavy problems which do not have to strictly meet timing constraints that exist in the real-time control applications. Instead, they try to utilize the parallel system via dynamic load balancing which results in complicated scheduling algorithms not necessarily guaranteeing the minimum calculation time because processes are migrating among processors. Burns and Wellings [18] propose that the next generation of transputers should have more task priority levels and via manipulation of the priority levels combined with the implementation of a suitable scheduler the deadline could be met. However, this is research directed towards real-time operating systems. The commercially available transputer operating systems are not real-time operating systems.

An additional disadvantage of the current transputer operating systems is that the host PC computer is relatively slow and all calculations are performed on the transputers while the PC is used only as a slave file server with the keyboard and monitor. This is not convenient for our application because the real-time control requires fast servicing of hardware interrupts and I/O boards. Although some software drivers for the I/O boards could be executed on the transputer network, this solution is slow and awkward because an I/O command from transputer must be first converted to serial signal, then converted into its parallel representation, read by the host 386 CPU via the transputer port, followed by DOS or bios call to execute the desired I/O operation. If the desired I/O operation is reading, there is an additional sequence of operations that must be executed. Such an approach is likely to have large overhead and does not solve the problem of the interrupt service routine initiated by the hardware interrupts on the I/O bus.

The cost of the software environment must be considered together with the fact that any programming language for the transputer network must be purchased separately. Cost and the previously mentioned considerations led to the decision to purchase a parallel C programming language designed by Logical Systems. The idea of buying an operating system or other software tools was abandoned. Instead, a new goal was set to develop a simple communication protocol which would enable full utilization of the host PC system and the transputer network.

This approach is not usually recommended because there is not sufficient information available about the Quadputer board interface and communication protocol to implement the connection between the PC I/O bus and the root transputer. Later it was discovered that it would be easier if software tools were purchased which could enable postmortem analysis and transputer monitoring. These tools are important for the low level software development.

The choice of parallel C programming language was influenced by the cost of the software development environment and need not to use an additional programming language. Since the Logical System Parallel C Toolkit as well as the Microsoft C v5.1 use the ANSI C, it is possible to develop a significant portion of the C code for the MIMO GPC on the host PC computer and then to port it to the transputer environment. However, porting the code is not straightforward when either the PC side is not used as a slave file server or when multiple transputers are used, i.e. the algorithm is parallelized. On the positive side, one can use all software tools available for Microsoft C and when the program modules are debugged they can be used with a high degree of confidence on the transputer system.

8.5 Message Passing Protocol

The Logical System Parallel C provides an option not to use the host PC computer as a file server. This is achieved, on the transputer side, by disabling command line arguments to be passed to the root transputer when the program is loaded to the transputer network and by disabling all transputer I/O library calls. On the host PC computer side, a user can specify a name of the program which will start after the transputer downloading is finished. Typically, this program has to handle communication between the host PC computer and the root transputer, provides file server services, and controls the I/O hardware.

There are several problems to solve when a simple message passing protocol is designed, for example, low level communication, message routing, dead-lock avoidance, etc. The low level communication which takes care of passing bytes between the host PC computer and the root transputer follows the guarded input principle as described by Jones [49], however, with the modification that the root transputer always initiates the information exchange with the host PC computer. The host PC computer does not initiate any communication due to the Quadputer board interface design. There are two ports in the host PC computer I/O space which indicate readiness of the root transputer to send or receive one byte. The crucial shortcoming is that the Quadputer board interface provides a one byte hardware buffer for reading a byte from the I/O bus. When the buffer is empty an appropriate bus port indicates that the the root transputer is ready to read a byte. However, an empty hardware buffer does not mean that the root transputer wants to receive a byte. This indication is not very helpful because the host PC computer does not know if the root transputer wants to receive a byte. The second port, indicating the readiness of the

root transputer to send a byte to the host PC computer I/O bus, carries true information, i.e. its content is true only and only when the root transputer wants to write a byte to the I/O bus. Note that the inter-transputer communication uses the link communication also for process synchronization. If a process made a commitment to send or receive a message, the process can continue only after the message is passed.

A message is characterized by several attributes. These attributes are typically the destination, source, type, length, and buffer address for each message. Although the source of the message is not necessary, it helps to determine specifics of a particular message and enables analysis of corrupted messages. The buffer address is specific for each processor and it does not make sense to send it with the message.

Since the message length can be different for each message and must be known to each processor, the following philosophy of the message passing protocol was implemented. First, there is a unique header file *message.h* included in each file controlling sending or receiving messages. This guarantees a consistent message system. This *message.h* file contains definitions of each message. The definition is one byte long and contains destination, source, and the type of the message. During the initialization phase of the program a data structure is created for each message which holds complete information about each message. For example, it contains length of the message, address of the message buffer, indication if the message buffer is empty or full, counter how many times the message driver tried to proceed with the message and its counterpart was not ready etc. The counter is helpful for debugging the host PC side to determine which message was initiated but never arrived.

In general, each message consists of two inseparable sequential communications. The first part is a one byte request message defining which message is requested. Each transputer can send this byte but the host PC computer can read it only due to the hardware limitations described previously. Once this byte is sent and received, both participants must dedicate the specified link to the message which belong to the one byte request message. If, for any reason, one side does not fulfill its receiving or sending responsibilities, this link is either frozen or the message system is corrupted.

A relatively wasteful approach is taken with respect to message buffers allocation to avoid message passing dead-locks and to detect a possible message corruption.

Since the software environment allows only a static allocation of parallelized algorithms, it is feasible to define routes for each message and allocate buffers on each processor through which a particular message passes. This corresponds to the cycle free message routes which according to Shumway [91] guarantees that the message

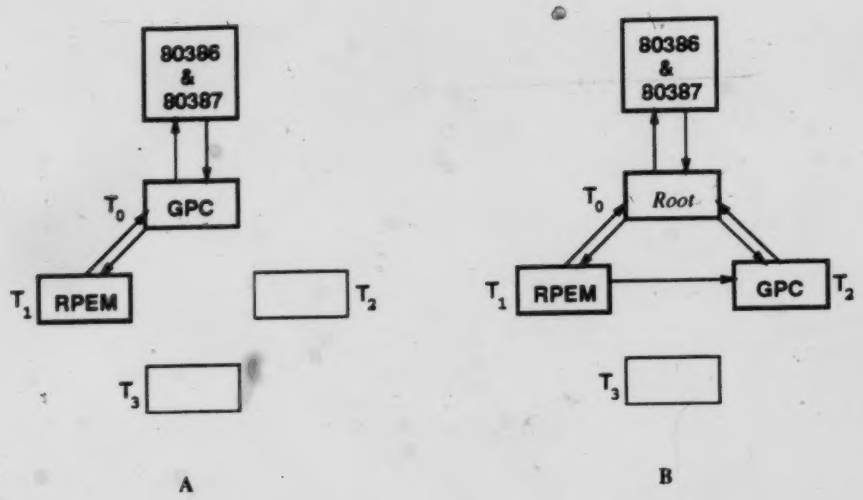


Figure 8.2: Examples of task allocation for 1D motion.

dead-lock does not occur as long as the messages are retrieved at their destination place. Additionally, if during passing through different buffers the message gets consistently corrupted, it is possible to trace it because each message is passing only through its buffers and its footprints are not corrupted by other messages. Also, some speed-up is gained with respect to the message passing system which uses circular buffers of constant size because the housekeeping is simpler.

There is another scheme for communication among processors. It consists of letting the root transputer be in charge of the entire message passing protocol. If the parallelized algorithm must be synchronized, it is relatively easy to control and synchronize the message flow by sending the one byte request messages to all remaining transputers and the host PC computer while utilizing the hardware parallelism of transputer links. This scheme is easier to manage and debug than a completely distributed control of message passing.

Other schemes of message passing can be used as a result of the design of this message passing protocol. The main disadvantage is that these schemes require more modifications whenever either the parallelization scheme or the system configuration changes. Two examples of possible communication schemes for 1D motion are shown in Figures 8.2A and 8.2B, where the arrow headed lines depict the message passing protocol. The configuration when the root transputer T_0 coordinates the entire message protocol and all remaining tasks are allocated to other transputers is shown in Figure 8.2B . Note that different task allocations require different topologies of the message system.

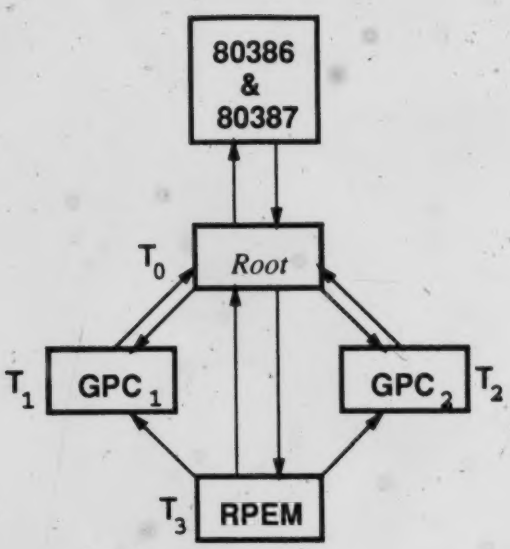


Figure 8.3: Example of task allocation for 2D motion.

Figure 8.3 shows an example of the task allocation and the associated message structure for full 2D decoupled motion. Although, the root transputer may not be fully utilized, the task allocation is easy to implement.

There is another consideration for data passing through the communication protocol between the host PC computer and the root transputer. The binary representation of numbers on the host PC computer is different from the binary representation on the transputer. Because the communication between the host PC computer and the root transputer is the communication bottleneck, the data is converted to the smallest binary representation to shorten the message length. For example, when 32 bit integers on the transputer system are converted to 16 bit integers needed on the PC system or 8 byte double precision floating point numbers are converted to 4 byte single precision floating numbers. Care must be taken of correct position of each byte because the binary representation also differs in the order in which bytes belonging to the same number are stored in the memory.

An additional set of conversion routines, with similar goals, was developed to convert the floating point numbers resulting, for example, from the final GPC calculations to the integer numbers representing the ± 10V range of DAC as well as converting integers from ADC to the floating point representation needed for the control and identification calculations.

The final note on the developed message passing protocol is related to its universality and efficiency. Obviously, the goal of having universal and extremely efficient

communication protocol is contradictory. In the case of implementation of real-time systems it is necessary to guarantee that requested messages are arriving at least on time. A message which is late may have disastrous effects for the entire system. The developed message passing protocol is not universal because the main goal is its efficiency and speed. The main disadvantage of this protocol is that whenever a new parallelization of the control algorithm or a new configuration of the transputer network is tried, it is necessary to restructure the message passing protocol. This means changes in buffer sizes and in the control and synchronization of the messages.

Although the transputer can performing some calculations and sending and receiving messages concurrently, the host PC computer can only work sequentially. To resolve the problem of keeping up with the message system and handling the hardware interrupts and I/O requests, the PC side software is driven by the combination of hardware interrupts from the timer and polling of the message system. The polling of the message system from the PC side is necessary because all messages can be properly initiated only from the root transputer. Although, the idea of abandoning a message which did not arrive within a specified time is attractive, it is not acceptable in the parallel C environment which strictly states that an identical number of bytes must be sent and received otherwise the link state is undefined. It is, therefore, necessary to ensure that either the message arrived within its specified time or to safely terminate the experiment.

8.6 MIMO GPC Parallelization

The majority of algorithms are presented in the sequential form because they are designed for single processor computers. As Stone [93] indicates, the main problem for general acceptance of multiprocessors is concurrent programming. It involves a search for decomposition of the sequential algorithm into parallel tasks which can run concurrently. Additional care must be taken of message passing and task synchronization. All this should be performed so that maximum advantage or speed-up can be achieved on a given multiprocessor configuration. Although parallelizing compilers has had some success for SIMD (single instruction multiple data) computers, the parallelization of MIMD computers is currently an area of active research. Note that many aspects must be considered. It is necessary to parallelize algorithms so that they respect algorithm correctness, processor performance, processor configuration, reasonable communication overhead, synchronization, optimum loading of processors, and a good use of I/O devices.

The situation becomes more difficult for real-time applications because hard time constraints must be met. When experiments are performed with different parameters directly influencing the duration of certain processes, there is a limit beyond which increasing certain parameters will result in timings beyond deadlines which is not permissible for real-time applications. Therefore, real-time applications must be able to detect such situations and safely terminate the experiment.

A typical problem in parallel processing is the allocation of tasks to specific processors so that the multiprocessor system can be fully utilized. For problems without hard time constraints, synchronization, message passing, and task granularity are the main variables influencing the overall processing time. The task or process granularity is defined as the ratio R/C where R is the execution time and C the communication time for a given process. It is generally concluded that large granularity means low utilization of the multiprocessor because it is very likely that some processors will be waiting for synchronization and communication. Small granularity means degraded performance because a lot of time is spent by communication. When real-time applications are considered, the ultimate decisive factor is meeting the strict deadlines.

Although it is possible to optimize the multiprocessor utilization for a given fixed application, it is not peformed for the adaptive MIMO GPC because the timing is a complicated function of control and identification variables and changes whenever these variables are modified. A small granularity also makes the algorithm more difficult to debug. Only very coarse parallelization is performed by separating the MIMO GPC and the MIMO RPEM, although the complete MIMO GPC and MIMO RPEM were also implemented together on the root transputer. The crucial role for this type of parallelization is synchronization and message passing. If a strict condition is imposed that the parameters updated by the last measurements must be used for the current control calculations then the following two scenarios, depicted by the corresponding data flow diagrams in Figure 8.4, are almost identical. Each arrow headed line in the data flow diagrams represents a message (data flow) and each oval, circle, and ellipse represents a process. Notice, that each process is indicated on the transputer where it will be executed. Each process can start its execution only when all messages arrived. Such a representation resembling data flow diagrams is possible because we deal with static allocation of processes to processors. A comparison of both alternatives shows that it is preferable to allocate all processes on the root transputer because this solution is faster and easier to debug.

If the condition for using the identified parameters from the latest measure-

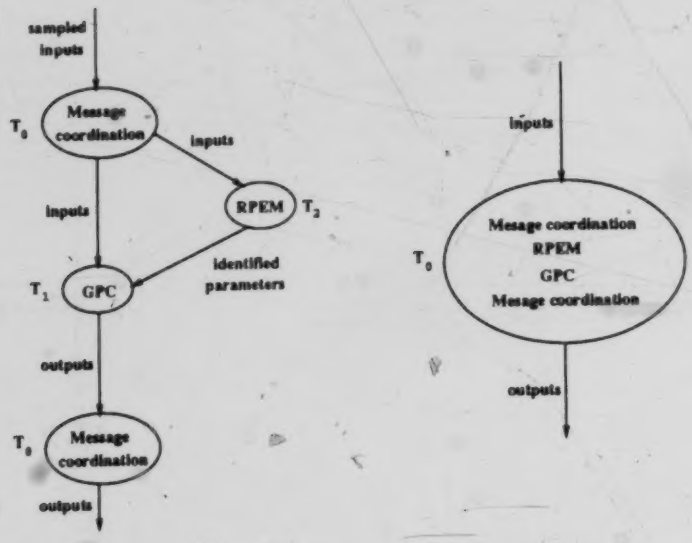


Figure 8.4: 1D adaptive GPC with strict timing requirements.

ments is relaxed to the point that the identification must be only synchronized within a sampling period, certain advantages can be obtained. This situation is shown in Figure 8.5. The dashed arrow indicates that a message will be accepted if it is available and, if not, the GPC will proceed with the last available identified parameters. Synchronization is achieved by sending the one byte request message for sending the identified parameters from RPEM to GPC node. The test of readiness of this byte from RPEM is performed before and after the MIMO GPC calculations so that the newest updated identified parameters are not missed. If no request from RPEM is available, the GPC task may decide to continue its work with the last available identified parameters. The basic advantage of this parallelization is that more time is available for the RPEM and GPC calculations allowing longer horizons and higher order models. The disadvantage is that at the n-th sampling period the $(n-1)$-st updates of the identified parameters are likely to be used.

When the 2D motion is considered, both motions can be decoupled as shown in the following schemes in Figure 8.6. This implementation is a replication of 1D motion using two independent motions. If only the latest updates are allowed, the right-hand scheme is used. If use of earlier identified parameters is permitted, the left-hand scheme is used with the advantage that more time is available for the GPC and RPEM calculations. In such a case the identification processes for both motions are allocated to the same transputer because the RPEM calculations take less time than the GPC calculations.

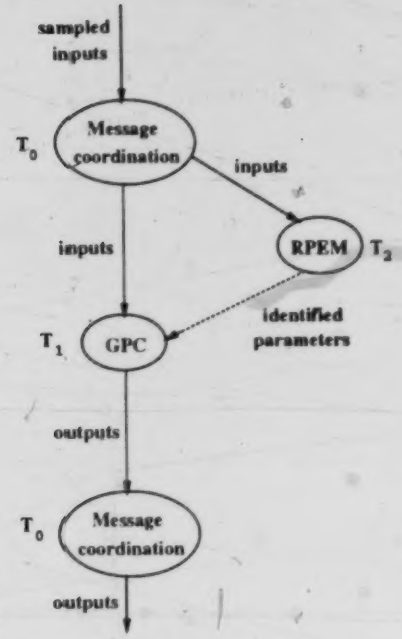


Figure 8.5: 1D adaptive GPC with relaxed timing requirements.

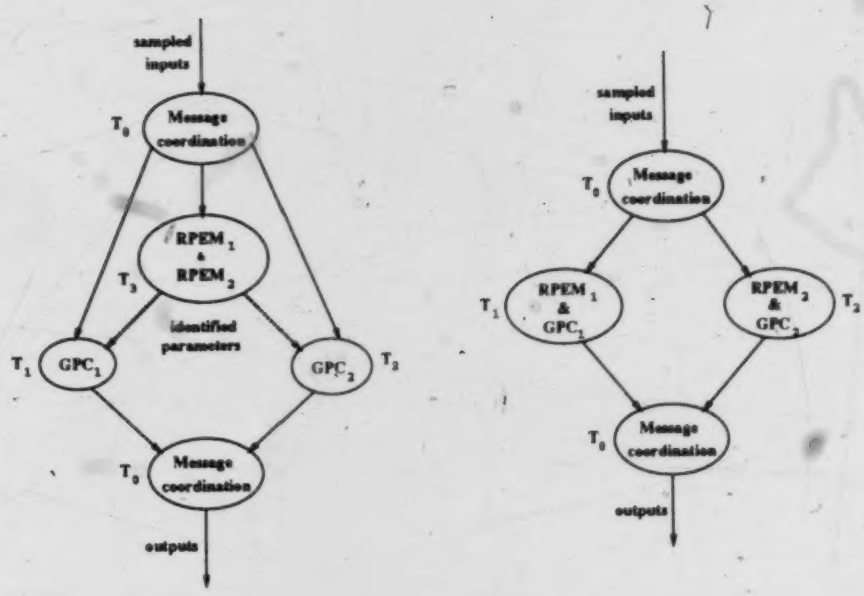


Figure 8.6: Adaptive GPC for decoupled 2D motion.

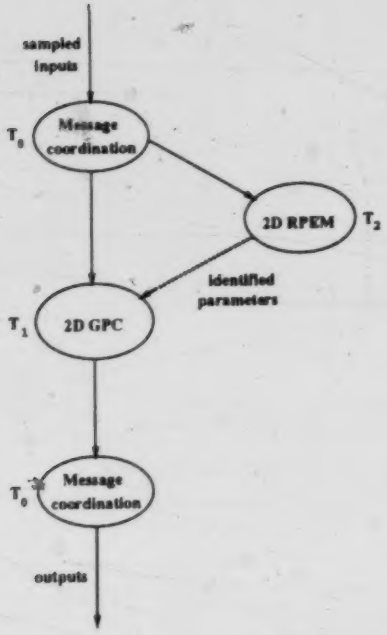


Figure 8.7: Adaptive GPC for coupled 2D motion.

If fully coupled 2D motion is considered, it is necessary to carry out the entire GPC on the same transputer and the entire RPEM on another transputer so that reasonable horizons can be achieved. This implies that it is necessary to allow the use of earlier identified parameters. This scheme is described in Figure 8.7.

Because the available hardware and software allow only static task allocation, choice of a particular scheme means programming modifications so that proper synchronization and message passing guarantee the algorithm correctness.

8.7 Debugging and Error Detection

There are several difficulties with implementing the parallel algorithms. First of all, no debugging tools are available, so it is necessary to use engineering intuition to deduce the cause of a complete failure of the algorithm. It is more difficult to discover errors which result in only a partially corrupted algorithm, i.e. the algorithm executes but the results are not correct. The main problem is that at this stage simulations cannot reproduce all intermediate results occurring when the algorithm is executed concurrently. It requires a considerable effort to implement the message passing protocol and the timing information into simulations so that simulations re-

sults can reasonably resemble the parallel execution results. Development of monitors for transputer parallel processing environment is an area active research. The monitors are rather intrusive because they influence timings of the monitored processes as is obvious from the work of Cai and Turner [19].

Considering the potential problems and recognizing that the flexible manipulator may be driven by a significant torque, the following heuristic safeguards are proposed. If the parallel algorithm is frozen, either a constant torque is required for a longer time period or no message, with the torque control signal, arrives by the deadline defined by the sampling period. If either case occurs then the experiment should be aborted on the PC side. This means to change the requested torque to zero and to display a message announcing termination of the experiment. This solution minimizes the mechanical impact on the flexible manipulator if it runs into its limit switches and the emergency brakes are activated. Another precaution is to set up software position limit switches so that if the algorithm generates, for example, a control signal which makes the system unstable and the motor shaft crosses the software limit positions, a torque of the opposite sign should be delivered for a short time followed by a zero torque. Afterwards, the experiment should be completely aborted, data files should be saved, and the user should be informed about this problem.

In conclusion, parallelization of algorithms for MIMD computers used for real-time applications is not an easy task that could be fully automated in the foreseeable future. The algorithm implementer-programmer must exercise common sense to parallelize and debug the algorithm so that the final parallel control algorithm is correct, fast, and safe.

8.8 Additional Speed Considerations

There are several considerations related to the speed-up of the overall parallel system, i.e. including the host PC computer, and speed of trying a new run of the algorithm with different settings.

If speed is the most important consideration, then the small memory model must be used for the program running on the host PC computer. The limitation of the small memory model is that the code memory segment and the data memory segment can be 64 Kbyte at maximum. Although 64 Kbyte of memory available for the code is usually more than adequate, 64 Kbyte of memory for the data is quite a limiting factor.

To minimize the communication time between the host PC computer and the

root transputer, the control signals are not sent back to the root transputer because they are available to the adaptive MIMO GPC, executed on the transputer network, from the previous sampling period. The periodic message from the root transputer to the host PC computer contains only the control torque in the PC integer form so that the host PC computer does not have to perform any conversions.

To evaluate the MIMO RPEM, the identified parameters must be stored at each sampling period so that they can be viewed after the experiment or simulation is completed. Although the parameters could be sent to the host PC computer during each sampling period, it is not advisable for the following reasons. First, either the message containing the identified parameters would have to be reprogrammed whenever the number of identified parameters is changed or the message length would have to be sufficiently large and constant so that all reasonable model variations could be accommodated. Both solutions are inefficient. Second, timing requirements of this message would not be negligible with respect to the sampling period when this messages is sent to the host PC computer, ie. 24 parameters $\times$ 4 bytes = 96 bytes and 100kbyte/sec suggest that in ideal case it would take approximately 1 msec. Third, the data storage requirements on the PC side for only 400 steps lead immediately to an about 40 Kbytes of data memory. Storing the identified parameters in a file during each sampling period is unrealistic because the operation takes approximately the entire sampling period and does not let the host PC computer work on other functions. Using a large memory model slows down the PC computer. The logical solution is to have all identified parameters stored in the memory of the transputer system so that when the control algorithm is finished, the identified parameters are sent to the host PC computer and stored in a file for later evaluation.

Since the adaptive MIMO GPC has many tuning knobs and parameters, a researcher is frequently interested in investigation of the effects of changes in the tuning knobs and parameters. If each change required a recompilation of the program, the frequency of experiments could be slowed significantly. For convenience, all parameters and tuning knobs are stored in a file on the host PC computer. After all processes have been downloaded to the transputer network the initial phase of the program running on the host PC computer reads this file and creates a message that is sent to the processes on the transputer network so that they can carry out their own initializations. This enables frequent turnarounds of the experimental runs and the settings of all parameters and tuning knobs that result in satisfactory performance without unnecessary delays.

8.9 Algorithm Transformations

The software development of the MIMO GPC and MIMO RPEM algorithms can be described by several phases. Whenever a new phase commences, only the basic core of the algorithms remained unchanged. Other sections need to be modified.

The first phase was development of basic subroutines in MATLAB as well as in C programming language. The parallel development of the subroutines in MATLAB and C provides the opportunity to test individual sections of both algorithms. Although it appears that this approach is wasteful, in reality it represents significant savings in the long run because it provides a facility to detect logical errors.

When the algorithms were extended towards the multi-input multi-output version, further development in MATLAB was not pursued. At this stage of software development MATLAB cannot efficiently implement the data structures and programming techniques used in C. Nevertheless, the experience showed that testing the MIMO GPC and MIMO RPEM on a system consisting of several identical row subsystems without coupling terms against the corresponding SISO system in MATLAB can reveal surprising implementation errors.

The difficult part is the related implementation of the MIMO GPC and MIMO RPEM in the transputer parallel processing environment. The development undergoes several programming transformations which depend on the processor configuration and allocation of both algorithms on this processor configuration.

The first stage is to integrate the MIMO GPC with a simulated plant so that closed-loop behavior can be simulated fully on the host 386 PC computer. The next stage is to repeat these simulations on the root transputer when the host PC computer is used as a file server. This is not difficult because the simulation C source code can be directly compiled by the parallel C compiler. The only inconvenience is that debugging facilities are not available and only intermediate results can be stored in a file to be analyzed later. If both simulation versions, i.e. the complete simulations on the host PC computer and complete simulations on the root transputer, yield identical results, the development of the next phase can begin.

This phase implements distributed simulations, i.e. the plant is simulated on the host PC computer and the MIMO GPC and MIMO RPEM are executed on the root transputer. This step involves significant source code transformations. First of all, the communication protocol must be used to pass the plant outputs and control signals between the plant simulated on the host PC computer and the control algorithm running on the transputer system. This involves many changes

of the original algorithm, data transformation by the communication protocol with exact knowledge about the data allocation in each communication buffer etc. At this stage only the input-output behavior of the control algorithm can be observed via the generated control signals and the plant outputs which can be saved in a file. Any discrepancy between the simulation runs carried out completely on the transputer system and the distributed simulations leads to a complicated debugging procedure. Typically, another message is developed to verify the correctness of the data passed between the transputer system and the host PC computer. If the problem is not in data passing, the algorithm and the plant simulation must be compiled together for debugging.

If the distributed simulations perform well, the plant simulated on the host PC computer must be replaced by the program which processes the hardware timer interrupts, samples inputs, sends control voltages to the DD drive analog interface etc. This program also implements the friction compensation. To avoid recompilation each time any parameters or tuning knobs are changed, the program reads all necessary information from the file *setupvar* and passes it through the communication protocol to the MIMO GPC and MIMO RPEM allocated on the transputer system so that the adaptive GPC can be properly configured. In order to verify that these data were passed correctly, they are sent back to the host PC computer and stored in a file for comparison with the original parameters.

It is clear that the transformation from the distributed simulations to the real implementation of the MIMO GPC and MIMO RPEM on the transputer parallel system requires many changes which may introduce errors which are difficult to locate.

Any algorithm transformation, especially if it involves either new messages among used processors, requires substantial programming effort and is error prone. The algorithm transformations should be minimized because the current programming environment does not have debugging facilities for the transputer parallel processing environment.

Chapter 9

Conclusion

9.1 Summary

There is a significant difference between applications of control algorithms in simulation runs and in an actual experimental test-bed. The actual experimental use gives the ultimate answer to the effectiveness of a given control algorithm for the specific application. Simulations, however, do play an important role since they can be used to study the features of a particular control algorithm in a safe and inexpensive manner. An experimental research project should consist of both simulation runs and actual experimental implementation.

The main contributions of this project are directed towards addressing some of the key issues in control of actual flexible manipulators. These are listed below:

- A test-bed consisting of a flexible manipulator and associated computer and interface subsystems was designed and built. Such a test-bed provides the opportunity to investigate the effectiveness of different control laws under real-life conditions.

- A well known adaptive control algorithm developed for SISO systems, namely GPC was extended to MIMO systems and applied to the flexible manipulator. Since the model of the flexible manipulator was not known, a MIMO identification procedure (RPEM) was used along with the MIMO GPC.

- The heavy computational requirements of the adaptive MIMO GPC algorithm were addressed by introducing a parallel processing environment based on transputers. As a prerequisite to the actual implementation of the adaptive MIMO GPC in this parallel processing environment, many issues related to commu-

nication between the host PC computer and transputers were addressed and solved. Several schemes for the parallelization of the control algorithm were proposed and some of them were successfully implemented.

By means of simulation studies, the ability of the adaptive MIMO GPC to control nonsquare systems with less inputs than outputs was studied. Output decoupled and output coupled cases were considered. The following observations were made:

- The algorithm worked satisfactorily if the steady state requirements were not contradictory.
- It was observed that T-filters have similar effects as the output error weighting coefficients but their choice is based on engineering judgement.
- The minimum control horizon NU_i can be smaller than the number of badly located poles for nonsquare systems.
- It was also demonstrated that the MIMO output coupled plants are more difficult to control in case of the plant/model mismatch and to achieve good control results, it is beneficial to overparameterize the diagonal terms $A_{ii}(q^{-1})$ and to underparameterize the coupling terms $A_{ij}(q^{-1})$.

The experiments were confined to the motion of the flexible link in the horizontal plane. In this mode, the manipulator is a nonsquare plant with one input (control voltage) and two coupled outputs, one the motor shaft position, which is relatively noise free, and the other the acceleration of the tip of the flexible link which is a very noisy signal. The task of controlling these two coupled outputs with different noise levels necessitated a compromise for the adaptive GPC. This was achieved by a suitable choice of the T-filters and the output error weighting coefficients. In the above experiments, use of the incremental version of the adaptive MIMO GPC with proper T-filters yielded smaller steady-state errors. The use of either T-filters or output error weighting coefficients or both made lower order parameterization possible. Surprisingly, reasonable responses were obtained when the output coupled system was parameterized as an output decoupled (diagonal) system.

Both the simulation studies and actual experimental implementation showed that there are many overlapping tuning knobs in the adaptive MIMO GPC and some of them may have to be chosen based on experience and engineering judgement.

The experimental runs with the multivariable PID algorithm provided a basis for arriving at guidelines for the controller settings. This algorithm damps out the flexible link vibrations better than the MIMO GPC, however the motor shaft position control is significantly better with the MIMO GPC.

9.2 Recommendation for Future Research

The availability of a test-bed which reflects most, if not all, real-life dynamical characteristics, opens up a wide horizon for further research into the control of the flexible manipulator with other control algorithms, simple and more complicated, adaptive and nonadaptive. Some of the specific issues, which are worthy of immediate attention are:

- Minimization of the number of identified parameters using partial knowledge of the system.
- Force control using strain gages.
- High performance robust control.
- Other identification schemes.
- Fail safe mechanism in software.

To extend the study for motion in two dimensions, the following *hardware development* is deemed necessary:

- A mechanical device which can be used to guarantee the same initial position for different experimental runs.
- Additional control of the motor brakes to minimize the danger of serious damage to the flexible manipulator.
- Installation of strain gages to detect the direction of 2D vibrations.

Bibliography

[1] T.E. Alberts, L.J.Love, E.Bayo, and H.Moulin. Experiments with end-point control of a flexible link using the inverse dynamics approach and passive damping. In *Procedings of the 1990 American Control Conference*, pages 350–355, San Diego, California, May 1990.

[2] Altera Corporation. *Altera EPLD Handbook.* 1985.

[3] Altera Corporation. *Altera Data Book.* 1987.

[4] H. Asada. A geometrical representation of manipulator dynamics and its application ot arm design. *A.S.M.E. Transactions, Journal of Dynamics, Systems, Measurement, and Control*, 105:131–135, 1983.

[5] H. Asada and H.Yamamoto. Torque feedback control of mit direct-drive robot. In *Proceedings 14-th Int. Symposium on Industrial Robots*, pages 663–670, Sweden, 1984.

[6] H. Asada and I.H.Ro. A linkage design for direct-drive robot arms. *A.S.M.E. Transactions, Journal of Mechanisms, Transmissions, and Automation in Design*, 107:536–540, December 1985.

[7] H. Asada and K.Youcef-Toumi. Decoupling of manipulator inertia tensor by mass redistribution. In *Procedings of the ASME Mechanisms Conference*, pages 1–6, 1984.

[8] H. Asada and K.Youcef-Toumi. *Direct-Drive Robots, Theory and Practice.* The MIT Press, first edition, 1987.

[9] H. Asada and T.Kanade. Design of direct-drive mechanical arms. *A.S.M.E. Transactions, Journal of Vibrations, Acoustics, Stress, and Reliability in Design*, 105:312–316, 1983.

[10] K.J. Astrom and B.Wittenmark. *Computer Controlled System, Theory and Design.* Prentice-Hall, Inc., first edition, 1984.

[11] C.G. Atkeson, C.H.An, and J.M.Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986. Fall.

[12] M.I. Barlow, P.Konnanov, and S.E.Burge. Analogue I/O strategies for transputers. *Microprocessors and Microsystems*, 13(6):387–395, July/August 1989.

[13] A.S. Beck and D.McCloy. A comparative study of the power requirements of several basic robot design. In *Proceedings 16-th Int. Symposium on Industrial Robots*, pages 77–86, Brussels, Belgium, September 1986.

[14] M. Bühler, L.L.Whitcomb, F.Levin, and D.E.Koditschek. A distributed message passing computational and I/O engine for real-time motion control. In *Procedings of the 1989 American Control Conference*, pages 478–483, Pittsburgh, PA, June 1989.

[15] G.J. Bierman. Measurement updating using the U-D factorization. *Automatica*, 12:375–382, 1976.

[16] G.J. Bierman. *Factorization Methods for Discrete Sequential Estimation.* Academic Press, first edition, 1977.

[17] W.J. Book and M.Majette. Controller design for flexible distributed parameter mechanical arms via combined state space and frequency domain techniques. *A.S.M.E. Transactions, Journal of Dynamic Systems, Measurements, and Control*, 105:245–254, December 1983.

[18] A. Burns and A.J.Wellings. Occam's priority model and deadline scheduling. In T.Muntean, editor, *Parallel Programming of Transputer Based Machines*, pages 146–159, Grenoble, France, September 1987. OUG-7, Proceedings of the 7th Occam User Group Technical Meeting.

[19] W. Cai and S.J.Turner. Highly transparent monitoring of real-time occam programs. In *NATUG-1, Proceedings of the First Conference of the North American Transputer Users Group*, pages 41–51, Salt Lake City, Utah, April 1989.

[20] R.H. Cannon and E. Schmitz. Initial experiments on the end-point control of a flexible robot arms. *Int. Journal of Robotics Research*, 3(3):62–75, 1984.

[21] C. Canudas, K.J.Astrom, and K.Braum. Adaptive friction compensation in d.c. motor drives. In *Proceedings 1986 IEEE Int. Conference on Robotics and Automation*, pages 1556–1561, San Francisco, California, April 1986.

[22] N.G. Chalhoub and A.G.Ulsoy. Control of a flexible robot arm: Experimental and theoretical results. *A.S.M.E. Transactions, Journal of Dynamics, Systems, Measurement, and Control*, 109:299–309, December 1987.

[23] D.W. Clarke. Self-tuning mutistep optimisation controllers. In *Workshop on Adaptive Control Strategies for Industrial Use*, pages 1–25, Lodge Kananaskis, Alberta, Canada, June 1988.

[24] D.W. Clarke and C. Mohtadi. Properties of generalized predictive control. *Automatica*, 25(6):859–875, 1989.

[25] D.W. Clarke, C. Mohtadi, and P. S. Tuffs. Generelized predictive control - Part I and II. *Automatica*, 23(2):137–160, 1987.

[26] J.J. Craig, P.Hsu, and S.Shankar Sastry. Adaptive control of mechanical manipulators. In *Proceedings 1986 IEEE Int. Conference on Robotics and Automation*, pages 190–195, San Francisco, California, April 1986.

[27] John J. Craig. *Introduction to Robotics Mechanics & Control.* Addison-Wesley, first edition, 1986.

[28] John J. Craig. *Adaptive Control of Mechanical Manipulators.* Addison-Wesley, first edition, 1988.

[29] R.M.C. De Keyser, G.A. Van de Velde, and F.A.G.Dumortier. A comparative study of self-adaptive long-range predictive control methods. *Automatica*, 24(2):149–163, 1988.

[30] J. Denavit and R.S.Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, pages 215–221, June 1955.

[31] D.D. Donalson and C.T.Leondes. A model referenced parameter tracking technique for adaptive control systems i and ii. *IEEE Transactions on Application and Industry*, 82(68):249–262, September 1963.

[32] D. Dubowsky and D.T.DesForges. Robotic manipulator control systems with invariant characteristics. In *Procedings of the Fifth World Congress on Theory of Machines and Mechanisms*, pages 1515–1518, Montreal, Canada, 1979.

[33] H. Durrant-Whyte. Practical adaptive control of actuated spatial mechanisms. In *Proceedings 1985 IEEE Int. Conference on Robotics and Automation*, pages 650–655, St. Luis, Missouri, March 1985.

[34] E. Freund. Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators. *The International Journal of Robotics Research*, 1(1):65–78, 1982. Spring.

[35] B. Gebler. Feed-forward control strategy for an industrial robots with elastic links and joints. In *Proceedings 1987 IEEE Int. Conference on Robotics and Automation*, pages 923–928, Raleigh, North Carolina, March 1987.

[36] W.B. Gevarter. Basic relations for control of flexible vehicles. *AIAA Journal*, 8(4):666–672, 1970.

[37] A. Gogoussis and M.Donath. Coulomb friction effects on the dynamics of bearings transmission in precision robot mechanisms. In *Procedings of the 1988 IEEE Int. Conf. on Robotics and Automation*, pages 1440–1446, Philadelphie, Pennsylvania, 1988.

[38] D.F. Golla, S.C.Garg, and P.C.Hughes. Linear state-feedback control of manipulators. *Mechanism and Machine Theory*, 16:93–103, 1981.

[39] T.J. Graettinger and B.H.Koch. The acceleration radius: A global performance measure for robotic manipulators. *IEEE Journal of Robotics and Automations*, 4(1):60–69, February 1988.

[40] I. Gustavsson, L.Ljung, and T.Soderstrom. Survery paper: Identification of processes in closed loop - Identifiability and accuracy aspects. *Automatica*, 13:59–75, 1977.

[41] G.G. Hastings and W. J. Book. Experiments in optimal control of a flexible arm. In *Procedings of the 1985 American Control Conference*, pages 728–729, Boston, MA, June 1985.

[42] G.G. Hastings and W. J. Book. Verification of a linear dynamic model for flexible robotic manipulators. In *Procedings of the 3rd IEEE Int. Conf. Robotics and Automation*, pages 1024–1029, San Francisco, CA, 1986.

[43] G.G. Hastings and W. J. Book. A linear dynamic model for flexible robotic manipulators. *IEEE Control System Magazine*, pages 61–64, 1987.

[44] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International (UK) Ltd, first edition, 1985.

[45] P. Horowitz and W.Hill. *The Art of Electronics.* Cambridge University Press, second edition, 1989.

[46] R. Horowitz and M.Tomizuka. An adaptive control scheme for mechanical manipulators - compensation of nonlinearity and decoupling control. *A.S.M.E. Transactions, Journal of Dynamic Systems, Measurements, and Control*, 108:127–135, June 1986.

[47] INMOS Limited. *Transputer Reference Manual.* Prentice Hall International (UK) Ltd, first edition, 1988.

[48] INMOS Limited. *Transputer Technical Notes.* Prentice Hall International (UK) Ltd, first edition, 1989.

[49] G. Jones. On guards. In T.Muntean, editor, *Parallel Programming of Transputer Based Machines*, pages 15–24, Grenoble, France, September 1987. OUG-7, Proceedings of the 7th Occam User Group Technical Meeting.

[50] A. Kathab. On adaptive control of robotic systems. In *Procedings of the International Symposium on Robot Manipulators: Modeling, Control and Education*, pages 329–332, Albuguerque, New Mexico, November 1986.

[51] P.K. Khosla and T.Kanade. Real-time implementation and evaluation of model-based controls on cmu dd arm ii. In *Procedings of the 1986 IEEE Int. Conf. on Robotics and Automation*, pages 1546–1555, San Francisco, California, 1986.

[52] A. Kiedrzynski. Mass-stiffness analysis of robot links. In *Proceedings 16-th Int. Symposium on Industrial Robots*, pages 151–158, Brussels, Belgium, September 1986.

[53] D.E. Koditschek. Natural motion for robot arms. In *Procedings of the 23-rd Conference on Decision and Control*, pages 733–735, Las Vegas, NV, December 1984.

[54] D.E. Koditschek. Adaptive strategies for control of natural motion. In *Procedings of the 24-th Conference on Decision and Control*, pages 1405–1409, Ft. Lauderdale , FL, December 1985.

[55] P. Kotnik, S. Yourkovich, and Ü.Özgüner. Acceleration feedback for control of a flexible manipulator arm. *Journal of Robotic Systems*, 5:181–196, June 1988.

[56] H. Krishnan and M.Vidyasagar. Control of a single-link flexible beam using a Hankel-norm-based reduced order model. In *Procedings of the 1988 IEEE Int. Conf. on Robotics and Automation*, pages 9–14, Philadelphie, Pennsylvania, 1988.

[57] T. Kubo, G.Anwar, and M.Tomizuka. Application of nonlinear friction compensation to robot arm control. In *Procedings of the 1986 IEEE Int. Conf. on Robotics and Automation*, pages 772–777, San Francisco, California, April 1986.

[58] R. Kulhavý. Restricted exponential forgetting in real-time identification. *Automatica*, 23(5):589–600, 1987.

[59] Martin Lambert. *Adaptive Control of Flexible Systems.* PhD thesis, University of Oxford, UK, 1987. Report No. OUEL 1707/87.

[60] C.L. Lindquist. *Active Network Design with Signal Filtering Applications.* Steward & Sons, first edition, 1977.

[61] L. Ljung. *System Identification: Theory for the User.* Prentice-Hall, Inc., first edition, 1987.

[62] L. Ljung and S.Gunnarsson. Adaptation and tracking in system identification - A survey. *Automatica*, 26:7–21, 1990.

[63] L. Ljung and T.Söderström. *Theory and Practice of Recursive Identification.* The MIT Press, second edition, 1987.

[64] J.Y.S. Luh. Conventional controller design for industrial robots - a tutorial. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:298–316, May/June 1983.

[65] J.Y.S. Luh, M.W.Walker, and R.P.Paul. On-line computational scheme for mechanical manipulators. *A.S.M.E. Transactions, Journal of Dynamic Systems, Measurements, and Control*, 102:69–76, 1980.

[66] J.Y.S. Luh, M.W.Walker, and R.P.Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, AC-25(3):468–474, 1980.

[67] J. Maples. *Force Control of Robotic Manipulator with Structural Flexibility.* PhD thesis, Stanford University, 1985. Dept. Electrical Engineering.

[68] L. Meirovitch. *Analytical Methods in Vibrations.* The MacMillan Company, first edition, 1967.

[69] L. Meirovitch. *Elements of Vibration Analysis.* McGraw-Hill Book Company, second edition, 1986.

[70] D.R. Meldrum and M.J.Balas. Application of model reference adaptive control to a flexible remote manipulator arm. In *Procedings of the 1986 American Control Conference*, pages 825–832, Seattle, WA, June 1986.

[71] D.R. Meldrum and M.J.Balas. Application of model reference adaptive control to a flexible remote manipulator arm: A summary. In *Procedings of the International Symposium on Robot Manipulators: Modeling, Control and Education*, pages 213–220, Albuguerque, New Mexico, November 1986.

[72] C. Mohtadi-Haghighi. *Studies in Advanced Self-tuning Algorithms.* PhD thesis, University of Oxford, UK, 1987. Report No. OUEL 1689/87.

[73] G. A. Montague, M. T. Tham, and A. J. Morris. Restricted exponential forgetting in real-time identification. In *Proceedings of the 1986 ACC*, pages 721–727, June 1986.

[74] J.J. Murray and C.P.Neuman. Arm: An algebraic robot dynamic modeling program. In *International Conference on Robotics*, pages 104–114, Atlanta, Georgia, March 1984.

[75] R.J. Mutha. Constrained long range predictive control. Master's thesis, University of Alberta, Fall 1990. Department of Chemical Engineering.

[76] NSK Motornetics Corporation. Private communication.

[77] H.A. Pak and P.J.Turner. Optimal tracking controller design for invariant dynamics direct-drive arms. *A.S.M.E. Transactions, Journal of Dynamic Systems, Measurements, and Control*, 108:360–365, December 1986.

[78] P.C. Parks. Lyapunov redesign of model reference adaptive control systems. *IEEE Transactions on Automatic Control*, AC-11(3):362–367, July 1966.

[79] R.P. Paul. *Robot Manipulators; Mathematics, Programming and Control.* MIT Press, first edition, 1981.

[80] V. Potkonjak and N.Jakšic. Contribution to a computer-aided choice of d.c. motors for manipulation robots. *Robotica*, 4:37–41, 1986.

[81] W.H. Press, B.P.Flannery, S.A.Teukolsky, and W.T.Vetterling. *Numerical Recipes in C, The Art of Scientific Computing.* Cambridge University Press, 1988.

[82] Eugene I. Rivin. *Mechanical Design of Robots.* McGraw-Hill New York, first edition, 1988.

[83] B.D. Robinson and D.W.Clarke. Robustness effects of a prefilter in generalized predictive control. *Preprint*, pages 1–20, October 1989.

[84] M.D. Rogers. Models and methods for recursive identification. Master's thesis, University of Alberta, Spring 1989. Department of Chemical Engineering.

[85] D. Schmitz, P.Khosla, and K.Kanade. In *Proceedings 15-th Int. Symposium on Industrial Robots*, pages 471–478, Tokyo, Japan, 1985.

[86] E. Schmitz. *Experiments on the End-Point Position Control of a Very Flexible One-Link Manipulator.* PhD thesis, Stanford University, 1985. Dept. of Aeronautics and Astronautics.

[87] H. Seraji. A new approach to adaptive control of manipulators. *A.S.M.E. Transactions, Journal of Dynamic Systems, Measurements, and Control*, 109:193–202, September 1987.

[88] S. L. Shah, C. Mohtadi, and D. W. Clarke. Multivariable adaptive control without a prior knowledge of the delay matrix. *System and Control Letters*, (9):295–306, 1987.

[89] Irving H. Shames. *Engineering Mechanics.* Prentice-Hall, Inc., third edition, 1980.

[90] D.S. Shook. *Implementation Issues in Adaptive GPC.* PhD thesis, University of Alberta, 1990. Department of Chemical Engineering.

[91] M. Shumway. Deadlock-free packet networks. In *NATUG-1, Proceedings of the First Conference of the North American Transputer Users Group*, pages 139–176, Salt Lake City, Utah, April 1989.

[92] G.S. Stiles. How the transputer stacks up to other processors: A comparison of performance on several application programs. In *NATUG-1, Proceedings of the First Conference of the North American Transputer Users Group*, pages 151–158, Salt Lake City, Utah, April 1989.

[93] H.S. Stone. *High-Performance Computer Architecture*. Addison-Wesly Publishing Company, first edition, 1987.

[94] Surgical and Medical Institute at the University of Alberta. Design of strain gage amplifier.

[95] L.M. Sweet and M.C.Good. Redefinition of the robot motio-control problem. *IEEE Control System Magazine*, pages 18–25, August 1985.

[96] K. Takase. Design of torque controlled manipulators composed of direct and low reduction ration drive joints. In M.Brady and R.Paul, editors, *Robotics Research, The First International Symposium*, pages 655–675. MIT Press, 1984.

[97] M. Takeshita, H.Segikuchi, T.Iwasaki, and T.Harima. Development of direct drive robot. In *Proceedings 15-th Int. Symposium on Industrial Robots*, pages 463–470, Tokyo, Japan, 1985.

[98] M.T. Tham, A.J.Morris, and G.A.Montague. Self-tuning process control: A review of some algorithms and their application. In *American Control Conference*, pages 996–1001, Minneapolis, MN, June 1987.

[99] C.L. Thornton and G.J.Bierman. UDU^t covariance factorization for Kalman filtering. In C.T.Leondes, editor, *Control and Dynamic Systems: Advances in Theory and Application*. Academic Press, 1980. Volume 16.

[100] M. Tomizuka. Adaptive control for robot manipulators. In *Proceedings of the Fifth Yale Workshop on Applications of Adaptive Systems Theory*, pages 188–193, Center for Systems Science, Becton Center, Yale University, May 1987.

[101] R.W. Toogood. Symbolic generation of robot dynamics equations, part I: The DYNAM/CLEAR system. Technical report, University of Alberta, Department of Mechanical Engineering, October 1987.

[102] T.T.C. Tsang and D.W.Clarke. Generalised predictive control with input constraints. *IEE Proceedings*, 6:451–460, November 1988.

[103] J.J. Uicker, J.Denavit, and R.S.Hartenberg. An iterative method for the displacement analysis of spatial mechanisms. *ASME Transactions, Journal of Applied Mechanics*, 17:309–314, June 1964.

[104] B. Wahlberg and L.Ljung. Design variables for bias distribution in transfer function estimation. *IEEE Transactions on Automatic Control*, AC-31(2):134–144, February 1986.

[105] R. Welburn. Ultra high torque motor system for direct drive robotics. In *Procedings of of the Robots 8 Conference - Applications for Today*, pages 19-63–19–71, Detroit, Mich., June 1984.

[106] L.L. Whitcomb and D.E.Koditschek. Robot control in a message passing environment: Theoretical questions and preliminary experiments. In *Procedings of the 1990 IEEE Int. Conf. on Robotics and Automation*, pages 1198–1203, Cincinnati, Ohio, 1990.

[107] D.E. Whitney. Resolved motion rate control of manipulators and human prostheses. MMS-10, Mich.(2):47–53, 1969.

[108] D.E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *ASME Transactions, Journal of Dynamic Systems, Measurements, and Control*, pages 303–309, 1972.

[109] D.E. Whitney, W.J.Book, and P.M.Lynch. Design and control considerations for industrial and space manipulators. In *Procedings 1974 Joint Automatic Control Conference*, pages 591–598, Austin, Texas, June 1974.

[110] K. Youcef-Toumi and H.Asada. The design of open-loop manipulator arms with decoupled and configuration-invariant inertia tensors. *ASME Transactions, Journal of Dynamic Systems, Measurements, and Control*, 109:268–275, September 1987.

[111] K.D. Young. Controller design for a manipulator using theory of variable structure systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8(2):101–109, 1978.

[112] K.D. Young. A variable structure model following control design for robotics applications. In *Proceedings 1986 IEEE Int. Conference on Robotics and Automation*, pages 540–545, San Francisco, California, April 1986.

[113] S. Yourkovich, A.P.Tzes, I.Lee, and K.L.Hillsley. Control and system identification of a two-link flexible manipulator. In *Procedings of the 1990 IEEE Int. Conf. on Robotics and Automation*, pages 1626–1631, Cincinnati, Ohio, May 1990.

Appendix A

Hardware Design

A.1 QDB

Block diagram and organization of the principal ORCAD files are shown in Figures C.2 and C.3 in Appendix C.

A.1.1 Quadrature Signal Decoder and Counter Subsystems

The inputs for the quadrature signal decoder subsystem are quadrature signals *channel A*, *channel B*, and signal *channel Z*. The *channel Z* signal is a marker pulse which occurs whenever the RDC counter generates a carry-out signal indicating that the RDC counter is full and starts counting from zero state again. This happens 150 times per each motor shaft revolution, when the motion reaches the boundary between two teeth of the DD motor. Jumper J2 provides an option to use the *channel Z* signal as a hardware interrupt. Presently, this option is not used.

The quadrature encoder TTL level signals *channel A* and *channel B* are led through a couple of inverters with Schmitt-trigger output to improve transition edges of the quadrature signals TTL levels. This is necessary in the case of slow motion of the shaft when the quadrature signals could stay in the undefined level region of TTL circuits for a longer period. Such improved signals are taken into the first EPLD900 (see Figure C.4 in Appendix C).

The digital prefilter was implemented as outlined in Hewlett Packard HCTL-2000 Quadrature Decoder/Counter Interface IC. It creates the input part of this EPLD900 and its function is to remove spike noise from the signals *channel A* and *channel B* to avoid false falling and raising edges of these signals. This is done via fast sampling of the input signals and accepting a change of logical level of any input

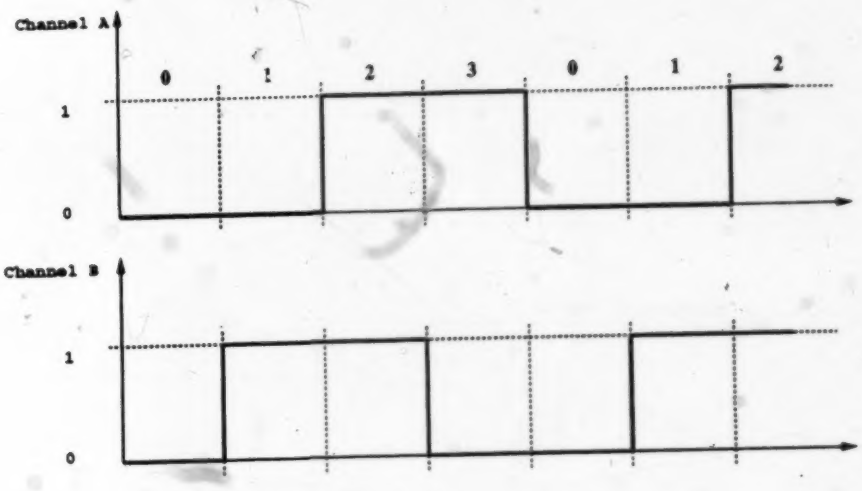


Figure A.1: Quadrature signals.

signal only if three consecutive samples at the opposite level exist. This raises a question of minimum sampling frequency for proper function of this filter. The worst scenario at the time of this design was as follows:

To guarantee good spike noise filtering, it is recommended that the digital filter sampling frequency sample the shortest pulse half wave at least 7 times. Because the motor can move with maximum velocity 5 RPS and each *channel A* or *channel B* signals provides 614400/4=153600 pulses/revolution, the digital prefilter sampling frequency must be larger than $5 * 153600 * 7 * 2 = 10.752$ MHz. Since the prefilter sampling frequency is derived from the I/O bus clock, direct implication is that the I/O bus clock should be more than 10.8 MHz. Although our 80386 based computer has the I/O bus clock frequency 8 MHz, it is sufficient because currently the DD motors are working in the mode at which maximum velocity is only 1.66 RPS.

The second part of the quadrature decoder subsystem located in this EPLD900 is the quadrature decoder circuit whose inputs are the filtered signals *channel A* and *channel B*, which are denoted A_n and B_n to indicate the latest sampled *channel A* and *channel B* signals. The reason is that it is necessary to distinguish previous A_{n-1} and B_{n-1} and present A_n and B_n signals to find the direction of a valid pulse.

Figure A.1 depicts typical quadrature signals and the associated states 0, 1, 2, and 3. The previous as well as present states of both signals can be described by their states.

STATE	A_n	B_n	A_{n-1}	B_{n-1}
0	0	0	0	0

$A_{n-1}B_{n-1}$ \ A_nB_n	00	01	11	10
00	0	UP	E	DOWN
01	DOWN	0	UP	E
11	E	DOWN	0	UP
10	UP	E	DOWN	0

Figure A.2: Karnaugh map for the QDB.

1	0	1	0	1
2	1	1	1	1
3	1	0	1	0

The following state transitions initiate counting up: $0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 0$. Counting down happens if transitions $3 \rightarrow 2, 2 \rightarrow 1, 1 \rightarrow 0, 0 \rightarrow 3$ occur. Any other transition is an error transition and should be discarded. If we define the following states: 0 = no counting, E = error and therefore no counting, UP = count up, DOWN = count down, we obtain the Karnaugh map of all possible state transitions shown in Figure A.2.

After some manipulations of logical identities, we obtain the following expression for signals UP and DOWN for valid counting transitions

$$UP = (A_{n-1} \oplus B_n)\overline{(B_{n-1} \oplus A_n)} \tag{A.1}$$

$$DOWN = \overline{(A_{n-1} \oplus B_n)}(B_{n-1} \oplus A_n) \tag{A.2}$$

Note, that the UP and DOWN signals stay on the logical level 1 only for one sampling period. It means that both signals have the form of short pulses that indicate which transition occurred. Both pulses are used for proper counter counting

after they are processed by additional logical circuits. First, a J-K flip-flop is used to generate an additional signal UP/DOWN that combines the information about the direction of counting. The UP and DOWN pulses are sufficiently delayed by three sampling periods to generate a signal COUNT, which always causes counter counting. These two signals UP/DOWN and COUNT are the output signals from the first EPLD900.

The second EPLD900 contains 24 bit up/down counters counting in $2's$ complement code. Although a commercial up/down 8 bit counter was taken as a template, when tested, it did not count properly when more than two such counters were cascaded to create a larger counter. The counter was redesigned to reflect the need for cascading and to fit into the macrocell topology of EPLD900 in the form of 6 bit counters using T flip-flops (see Figures C.5 and C.6 in Appendix C).

It should be pointed out that both EPLD900s can be cleared simultaneously by the same signal generated by the RESET command whenever initialization is requested.

A.1.2 Sampled Counter Memory and I/O Bus Drivers

To obtain correct information about the position of both motor shafts, it is necessary to ensure that both counters are read at the same time instant, when the SAMPLE command is executed from the control program. It is especially important when velocities and accelerations of the shafts are calculated numerically using the shaft position measurements. A set of tri-state output positive edge triggered D flip-flops 74HC374 reads the contents of both counters at the sampling instant and then serves as a memory of the sampled position. This permits reading of the shaft position when the READ command is issued, with accurate information about the shaft position at the sample instant.

Although the instantaneous values of both counters are being held in the tri-state D flip-flops, their outputs are not placed directly on the PC I/O bus. Instead, when a specific set of flip-flops is addressed, its outputs have to go through 74LS245 octal bus transceiver IC to guarantee proper logical signal levels on the I/O bus. This happens when the READ command is executed. The sampled counter memory and the I/O bus drivers are shown in Figures C.7, C.8, and C.9 in Appendix C.

A.1.3 Quadrature Board Controller

The controller of the QDB is completely integrated in EPLD1800 chip (68 pin PGA). It decodes and generates all control signals such that the QDB complies with the requirements for interfacing to the I/O bus and decodes and carries out the following instructions from the 80386 CPU:

- Clear and initialize the QDB.
- Sample and store current shaft positions in 32 bit 2's complement.
- Place a content of the specified 8 bit sampled counter memory chip 74HC374 on the data bus so that it can be read by CPU.

Since the QDB is in the PC I/O address space, only 8 bit data can be transferred between CPU and the addressed port by one I/O read or write instruction. The QDB base I/O address is 0x200. Addresses allocated for particular instructions are :

WRITE:	0x200	clear and initialize the QDB	
	0x201	sample and store shaft positions	
READ:	0x200	read bits	D7-D0 of the base motor shaft
	0x201		D15-D8
	0x202		D23-D16
	0x203		D31-D24
	0x204	read bits	D7-D0 of the shoulder motor shaft
	0x205		D15-D8
	0x206		D23-D16
	0x207		D31-D24

The EPLD1800 must recognize when any I/O instructions is addressing the QDB. This means that the lower 10 address bits must be carrying the address of the QDB and IOR (I/O read) or IOW (I/O write) I/O bus signals must be active at the same time. Only when this occurs, the instruction decoding logic evaluates which QDB instruction should be carried out. The base address of the QDB, which is 0x200, can be changed via setting switches in case an I/O address collision occurs with other I/O devices. Timing of I/O bus signals is critical and addressing of each 8 bits of the sampled counter memory must be delayed by one sampling period using *D* flip-flops. Without this delay the timing requirements are not satisfied and the CPU cannot read information from the QDB.

The circuit for generation of wait states is also a part of the EPLD1800 and the number of the inserted wait states is controlled by external jumpers. The controller schematic diagram is shown in Figures C.10 and C.11 in Appendix C.

A.2 DI/O Box

An H11L1 opto-coupler with Schmitt-trigger output was chosen because of its versatility with respect to operating voltages and speed considerations. The H11L1 opto-coupler is used for all digital inputs and outputs passing through CN2 and CN5 motor amplifier connectors except for the quadrature signals whose frequency range is beyond the H11L1 frequency specifications (see basic circuits in Figures C.12, C.13, and C.14 as well as the power supplies in Figures C.15 and C.16 in Appendix C).

All signals coming from the motor amplifier are converted via the H11L1 to the required TTL levels. Digital output signals coming from the computer are driven by the Data Translation DT2817 digital I/O PC board which is configured as 32 bit digital outputs. This board was chosen because it uses LSTTL ICs with high fan-out guaranteeing that each output can provide sufficient current for switching the H11L1.

The digital signals from the motor amplifier, after the conversion to the TTL levels, are read by the DI/O subsystem of the Scientific Solution Lab Master DMA PC board.

Two digital I/O boxes have been designed and built, one per each motor amplifier, including their own set of power supplies. Each digital I/O box contains 17 input interface circuits for CN5, 12 input and 8 output interface circuits for CN2 connector.

A.3 Strain Gage Amplifiers

The design of the strain gage amplifier (see the strain gage amplifier in Figure C.17 its the power supply in Figure C.18 in Appendix C) was modified to fit the requirements of this project. A voltage follower is inserted between the instrumentation amplifier and the LED Wheatstone bridge balance indicating circuit to obtain measurements not influenced by flipping of the indicator circuit. The set of jumpers is prepared to use different gains of the INA110 instrumentational amplifier. The entire strain gage amplifier box contains 10 individual strain gage amplifiers and each amplifier has its own 6 VDC voltage stabilizer to excite its strain gage. An additional jumper is provided to select either a single or double strain gage type of measurement. The

front panel provides access to a potentiometer for balancing the Wheatstone bridge to improve repeatability of measurements.

To minimize effects of ambient temperature on the measurements, 3 wire connection of strain gages to their Wheatstone bridges is used. The voltage output from the Wheatstone bridge is in the range of several mV and, therefore, noise may play a significant role in the measurements.

Two test circuits have been designed and built to test the strain gage amplifier with either single or double strain gages.

Let the displacement due to vibrations in the ${}^2Y$ direction be denoted $v(x,t)$. Then the measured strain of the flexible link close to the motor shaft is

$$\epsilon_{max} = \frac{a}{2}\frac{d^2v}{dx^2}_{|x=0} \tag{A.3}$$

where a is the side of the square cross-section, and x is the x coordinate measured along the ${}^2X$ direction.

A.4 4-th Order Bessel Antialiasing Filters

A normalized fourth order Bessel filter [60] is given by two cascaded second order systems with damping ratios and undamped natural frequencies: $\omega_1 = 1.6043, \zeta_1 = 0.6207, \omega_2 = 1.4310, \zeta_2 = 0.9580$. This describes the fourth order Bessel filter with the corner frequency equal to 1 rad/sec.

The design of antialiasing filters is very likely to require the corner frequency which is different from 1 rad/sec. Therefore, if the desired corner frequency is $\omega_c \neq 1$, then the resulting filter transfer function is given by

$$G_{Bes-IV}(s) = \frac{(\omega_c\omega_1)^2}{s^2 + 2s\zeta_1(\omega_c\omega_1) + (\omega_c\omega_1)^2}\frac{(\omega_c\omega_2)^2}{s^2 + 2s\zeta_2(\omega_c\omega_2) + (\omega_c\omega_2)^2} \tag{A.4}$$

When the corner frequency is chosen, a question arises about attenuation of the antialiasing filter at the folding frequency. This is answered by the Figure A.3 which shows attenuation as a function of the normalized frequency.

An additional issue related to the choice of the antialiasing filter is the time delay introduced to all frequency signal components up to the filter corner frequency. The average phase delay is defined as $\tau_p = -\alpha/\omega$ and the instantaneous phase delay, i.e. group delay, as $\tau_g = -d\alpha/d\omega$, where α is the filter phase shift. In case of linear phase filters, we have $\tau_p = \tau_g$. The Figure A.4 shows that up to the corner

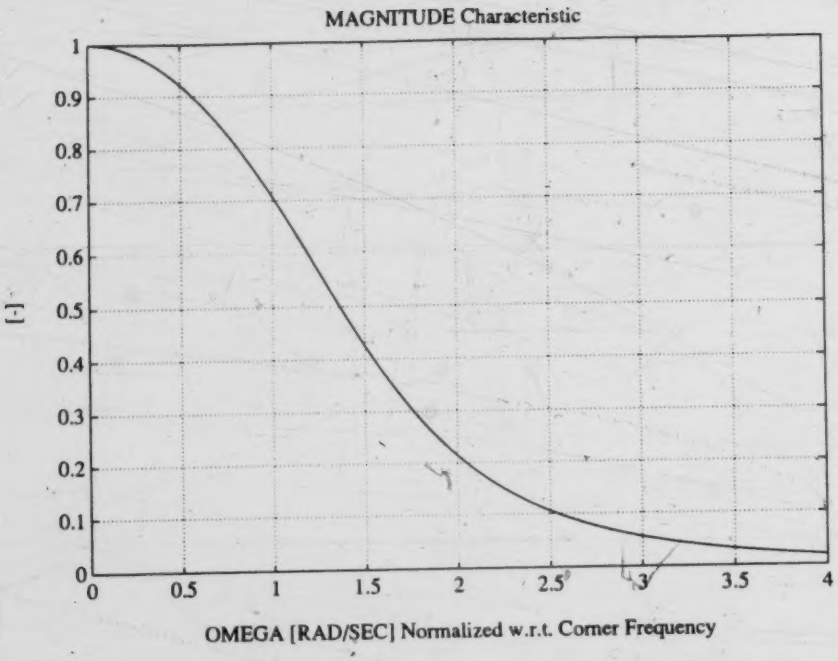


Figure A.3: Amplitude characteristic for 4-th order Bessel filter.

frequency the phase shift can be considered linear and at the corner frequency the phase shift is approximately $-120°$, i.e. $-2\pi/3$. Therefore the time delay up to the corner frequency is

$$\tau = \frac{2\pi/3}{\omega_c} \tag{A.5}$$

While the above expression provides absolute time delay in seconds, in control applications we would like to know by how many sampling periods T_s this signal will be delayed. Let us express the ratio τ/T_s as follows

$$\frac{\tau}{T_s} = \frac{\frac{2\pi}{3\omega_c}}{\frac{1}{f_s}} = \frac{\frac{2\pi}{3\omega_c}}{\frac{2\pi}{\omega_s}} = \frac{\omega_s}{3\omega_c} = \frac{2\omega_f}{3\omega_c} \tag{A.6}$$

where f_s is the sampling frequency and ω_f is the folding frequency. The above expression shows that as long as $\omega_f \leq 3\omega_c$ the delay of 2 sampling periods must be considered due to the antialiasing filter. Figure A.4 shows that for frequencies higher than the corner frequency a smaller delay occurs, i.e. they propagate through the filter faster but with smaller amplitude.

While so far only the theoretical issues of the design of Bessel antialiasing

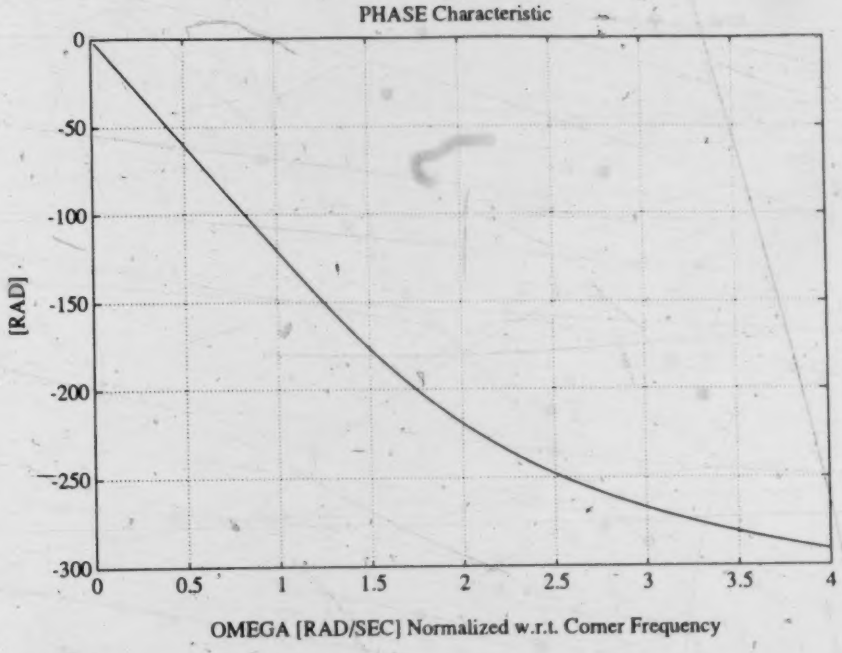


Figure A.4: Phase characteristic of 4-th order Bessel filter.

filters have been discussed, the implementation issues are equally important.

When facing a design of a filter whose corner frequency is not exactly known, it is desirable to use an approach which permits easy changes of the corner frequency without any substantial hardware modifications. Effects of component tolerances on the corner frequency as well as on the damping ratio should be considered to ensure a good quality filter even with imperfect components.

Two approaches were considered: switched-capacitor filters and classic active filter designs. After the comparison of the switched-capacitor filters and the classic active filters, a decision was made to develop and build the classic active filters with Bessel filter characteristics. There are [60] many possibilities to implement low-pass filter second order transfer function. The final choice is influenced by the following requirements:

- Provision must be made for changing the corner frequency in the future when the desired corner frequency will be specified.
- Number of components should be minimized.
- The filter should have small ω_c and ζ sensitivities to component variations.

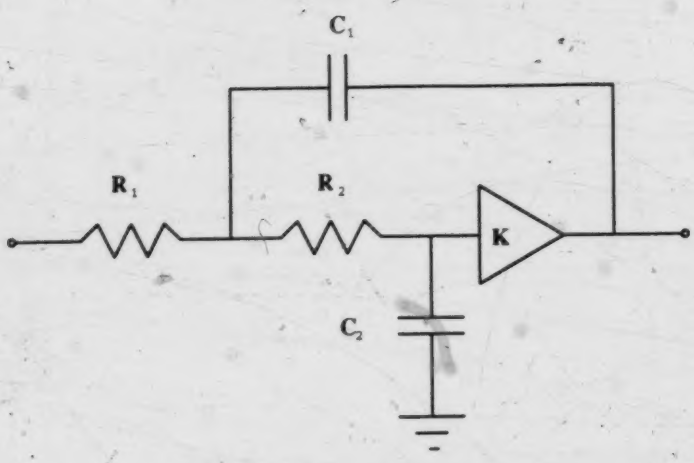


Figure A.5: Type 2B low-pass filter.

Advantages of the discrete component active filters with respect to switched-capacitor filters are, for example, ± 15 power supply, minimum DC offset with no need to attenuate and amplify signals, no need for additional antialiasing filter, signal amplitude $\pm 10V$, etc.

The requirements for the antialiasing filters can be satisfied even when additional conditions are imposed, e.g. design that uses the same capacitors for different corner frequencies. The circuit depicted in Figure A.5 is characterized by Lindquist [60] as 2B low-pass filter which is taken as the initial step of the design and is further modified. The transfer function of this filter

$$H(s) = \frac{K}{s^2 R_1 C_1 R_2 C_2 + s(R_1 C_1 (1 + C_2/C_1 - K) + R_2 C_2) + 1} \tag{A.7}$$

which can be seen as

$$H(s) = K \frac{1}{s^2/\omega_n^2 + 2s\zeta/\omega_n + 1} \tag{A.8}$$

To achieve small sensitivities, the following choices are made

$$R_1 C_1 = R_2 C_2 \text{ and } \quad \rho = C_2/C_1, \text{ i.e. } C_2 = \rho C_1 \tag{A.9}$$

and the transfer function can be rewritten as

$$H(s) = K \frac{1}{s^2 R_1 C_1 R_2 C_2 + s R_1 C_1 (2 + \rho - K) + 1} \tag{A.10}$$

with

$$\omega_n^2 = R_1 C_1 R_2 C_2, \qquad \zeta = (2 + \rho - K)/2 \tag{A.11}$$

Typical design procedure is as follows:

Given: ζ and ω_n

Choose: C_1, ρ such that $\rho < 1$

THEN:

$$R_1 = \frac{1}{\omega_n C_1}, C_2 = \rho C_1, \qquad R_2 = \frac{R_1 C_1}{C_2} = \frac{R_1}{\rho}, \qquad K = 2(1 - \zeta) + \rho \tag{A.12}$$

The sensitivities are

$$S_{R_1}^{\omega_n} = S_{R_2}^{\omega_n} = S_{C_1}^{\omega_n} = S_{C_2}^{\omega_n} = -\frac{1}{2} \tag{A.13}$$

$$S_K^{\zeta} = (1 - (2 + \rho)/2\zeta) \tag{A.14}$$

$$S_{R_1}^{\zeta} = -S_{R_2}^{\zeta} = (1 - 1/\zeta)/2 \tag{A.15}$$

$$S_{C_1}^{\zeta} = -S_{C_2}^{\zeta} = (1 - (1 + \rho)/\zeta)/2 \tag{A.16}$$

An important observation about sensitivities is that they are very reasonable and the corner frequency sensitivities are constant.

Since the 4-th order Bessel filter consists of two cascaded filters with their damping ratios $\zeta_1 = 0.9580$ and $\zeta_2 = 0.6207$, then their gains are

$$K_1 = 2(1 - \zeta_1) + \rho_1 = 0.084 + \rho_1 \tag{A.17}$$

$$K_2 = 2(1 - \zeta_2) + \rho_2 = 0.7586 + \rho_2 \tag{A.18}$$

The optimum choice ρ_1 and ρ_2 with respect to SNR would be such that K_1 and K_2 are close to 1 because the signal amplitude is not significantly changed by each second order transfer function. The choice of ρ_1 and ρ_2 is not governed only by the above gain considerations, it must also respect available components, mainly capacitors. The resulting implemented compromise is

$$C_{11} = 1\mu F \qquad \rho_1 = 0.47 \qquad \Rightarrow C_{12} = 0.47\mu F \tag{A.19}$$

$$C_{21} = 1\mu F \qquad \rho_2 = 0.1 \qquad \Rightarrow C_{22} = 0.1\mu F \tag{A.20}$$

This is an acceptable compromise, which for corner frequencies in the range $< 5, 200 >$ Hz, requires reasonable values of resistors R_{11}, R_{12}, R_{21}, and R_{22}. The resulting gains

are

$$K_1 = 0.5540 \text{ and } K_2 = 0.8586 \quad \Rightarrow K_3 = 1/(K_1 K_2) = 2.1023 \qquad \text{(A.21)}$$

Since the gains K_1 and K_2 are parts of characteristic equation of each filter, they have direct effect on ζ_i and ω_{n_i}. Therefore these gains should be realized as accurately as possible. From the previous choices of ρ_1 and ρ_2 K_1 and K_2 smaller than 1 are obtained. Crude implementation would be via resistor voltage dividers but this would load the filters and the filter parameters would change. If the resistor voltage dividers were located at the input to the operational amplifier, the analysis would be more complicated. It is also very likely that the desired feature of fixed capacitors for variable corner frequency would be lost. It would be inconvenient and expensive if all components need to be changed with each change of the desired corner frequency.

Therefore, a compromise was implemented in the form of resistance voltage dividers followed by voltage followers to obtain impedance separation. The final stage implements K_3 so that the overall gain is equal to 1.

Note, that C_1, C_2, R_{x1}, R_{x2} and ζ remain unchanged for different ω_n. Only R_1 and R_2 must be replaced with each change of ω_n. The same applies to the second transfer function. Therefore, whenever the corner frequency must be changed, only 4 resistors must be replaced. To facilitate this conveniently, these 4 resistors are soldered on a component carrier which is inserted into the appropriate socket on the filter printed circuit board. Thus, if resistors for particular corner frequency are already placed on the component carrier, it takes only a few seconds to accomplish the change of the corner frequency. This also implies the possibility of having different corner frequencies for different signals. The schematic diagram for the designed and built 4-th order Bessel antialising filter is shown in Figures C.19 and C.20 in Appendix C.

The choice of $R_1/R_2 = \rho_1$ and $R_3/R_4 = \rho_2$ should be performed such that the desired ratios are as accurate as possible with the initial constraint in the form of $R_1 = 1/(\omega_{n_1} C_1)$ and $R_3 = 1/(\omega_{n_2} C_3)$. Typically, the values of R_1 and R_3 are such that they are not equal to standard values of resistors with 1% or 2% tolerance; either smaller or larger resistors are available. To find the best compromise values of R_1, R_2, R_3, and R_4 from available resistors with standard values, a program has been written which requires only the desired corner frequency as its input parameter and provides the best choice of all four resistors from both 1% as well as 2% tolerance set.

Each filter is implemented on its own printed circuit board with edge con-

nectors so that it can be vertically inserted into the mother-board which provides ±15VDC and GND distribution for each filter.

The printed circuit boards for this project have been designed with ORCAD/SDT and ORCAD/PCB software packages. The author manufactured all boards using facilities available in his home Electrical Engineering Department. Although the design of printed circuit boards deserves a longer description, it will be only mentioned here that to obtain proper Postscript files for positive Mylar transparencies, it was necessary to do some programming changes in Postscript programming language and that entire process is not straightforward.

A.5 Design of the Double Integrator

To implement the double integrator shown in Figure 3.10, the following design requirements are defined :

- A good quality operational amplifier must be used.
- A low-leakage current feedback capacitor must be used.
- Discharging time of the feedback capacitor must be minimized.
- TTL voltage level signal from the computer must directly control the discharging process.
- Loading of the integrator outputs must be minimized.

The analysis of the integrator circuit [45] with grounded inputs to evaluate effects of the output offset voltage and the input bias current gives the following equation for the integrator output

$$V_o = V_{off} + (\frac{V_{off}}{RC} + \frac{I_{bias}}{C})t \qquad (A.22)$$

The above equation shows that the integrator integrates even with zero input signal due to operational amplifiers imperfections in the form I_{bias} and the output offset voltage V_{off}. OP77 bipolar precision operational amplifier was compared with a FET operational amplifier and the comparison showed that OP77 residual drift effects were smaller.

Quality of the analog integrator also depends on the leakage resistance of the feedback capacitor used because the integrated voltage continuously drops due

to gradual discharge through its leakage resistance, i.e. a resistor parallel with the feedback capacitor. Metallized polypropelene $10\mu F$ capacitors were chosen because their leakage resistance is larger than $30000M\Omega$ and they have a good temperature stability. The larger value of the capacitor provides an option to obtain the integration constant $k = 1/(RC)$ from $0.1(R = 1M\Omega$ and $C = 10\mu F)$ to $100(R = 10k\Omega$ and $C = 10\mu F)$ with standard values of resistors.

If the integrator output reaches the saturation level then the output signal does not provide any sensible information. To avoid saturation, two approaches can be taken (individually or combined) to overcome the associated problems:

- The integration constant $1/RC$ can be so small that the integrator never reaches its saturation level for a given application. This can result in poor resolution, i.e. the output signal will be very small so that small changes in velocity or position will not be detected by the 12 bit ADC.

- A larger integration constant can be implemented if, at the time when the integrator output is approaching the saturation, the following sequence of operations is carried out :

  - The integrator is sampled and the sampled value is stored in a variable.
  - The integrator is discharged and starts to integrate from $0V$ again.
  - The consequent samples of the integrator output are increased by the value stored in the previously allocated variable.

The implications of the proposed solutions to the saturation avoidance will be investigated in the next chapter. The process of discharging is not instantaneous and it is necessary to examine its timing. Knowing what inaccuracies are introduced by imperfect discharging process is of interest. Answers to these questions can be found only when parameters of the switches are known.

There are several considerations for the choice of the double integrator switches. The voltage to be discharged is in the range from $-10V$ to $+10V$ and the TTL signals are in the range from $0V$ to +4V. This calls for a CMOS (complementary MOSFET) analog switch with voltage level shifter so that the TTL signals control directly all switches. The DG211 and DG243 CMOS analog switches are chosen to simplify the interfacing between the computer and the double integrator. Both DG211 and DG243 provide a switch configuration that can be controlled by the same TTL computer signal.

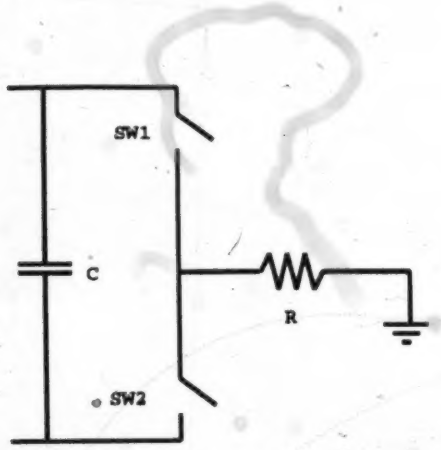


Figure A.6: Schematic diagram for discharging the double integrator.

To minimize the small leakage current going through the closed discharging analog switches, a special T circuit consisting of two analog switches and $1M\Omega$ resistor has been implemented to suppress the leakage current effects on the accuracy of the integration. The simplified schematic diagram of this T circuit is in Figure A.6 The resistance R_{on} of the closed analog switch used is about 105Ω. Because the value of the feedback capacitor $C = 10\mu F$, the discharging time constant is $\tau = 2R_{on}C = 2.1$ msec. This suggests that if the sampling frequency is smaller than $100Hz$ then one sampling period guarantees complete discharge of the integrator output voltage.

There are two reasons for using two switches in series. First, the analog switch leakage current effect on the accuracy of the integration is minimized. Second, if only one switch were used then the current flowing through the closed switch is $10V/105\Omega = 94.5$mA which is beyond the switch maximum rating. A compromise using only one analog switch is feasible if the switching output voltage is selected around 5V.

Before the output from each integrator goes to the double integrator board connector, it passes through a voltage follower to minimize the loading effects on the integration accuracy (see Figure C.21 in Appendix C).

There are two double integrator boards with edge connectors so that they can be inserted vertically on the same mother-board which hosts eight 4-th order Bessel filer boards. Both velocity and position can go via jumpers through Bessel filters if antialiasing is required. The design of the printed circuit board and considerations that went into building the double integrator boards are not described here.

A.6 Hardware Timer Interrupt Generation

Certain decisions must be made in advance to minimize potential problems. Originally, the IRQ2 (hardware Interrupt Request #2) was used and whenever errors were encountered, it was necessary to reboot the computer because the same interrupt is used to cascade an additional programmable interrupt controller 8259. The second 8259 uses the IRQ2 and its interrupt vector to report that its hardware interrupts are active and ask for CPU attention.

When the software driver for the hardware interrupts generated by 9513A works correctly, the final phase of the software driver renews the original interrupt vector for IRQ2 and unmasks all hardware interrupts so that the computer state is restored. If, however, the software driver for 9513A contains some logical errors then the interrupt vector may not be restored properly meaning that all devices using the cascaded 8259 interrupt controller do not have access to the hardware interrupt services. When this happens, then, for example, files and programs are not accessible because the hard drive controller hardware interrupts are not answered by the 80386 CPU.

Although most of the hardware interrupts on the first 8259 hardware interrupt controller are allocated to specific devices, IRQ5 can be used without danger of freezing essential functions of the CPU if the software driver is not completely debugged. After this change the time for development of timer interrupt generation was used more efficiently.

Only basic concepts of the hardware timer processing are discussed here. To minimize interference from other hardware devices, all hardware interrupts except for the timer and keyboard are masked. The interrupt vector belonging to the IRQ5 is preserved during the initialization so that it can be restored when the program finishes. The new interrupt vector is pointing to the assembler routine *irq5.asm* (see B.3 in Appendix B) which processes the hardware interrupt from the 9513A. When the 9513A generates an interrupt, the present program flow is interrupted and the program control is transferred to the routine *irq5.asm* which takes care of correct processing of hardware interrupts on the lowest level and also calls a C program that sets appropriate indications about the hardware signal which initiated the entire sequence. Afterwards the program control is returned to *irq5.asm* which enables interrupts and finally the program control is returned to the program which was originally interrupted by the IRQ5.

The file *tecmar.c* in the Appendix B.1 also contains routines which are used

when the 9513A is programmed to measure time of any program sequence with accuracy of 1μsec.

A.7 ADC Software driver

Since the Lab Master PC board has 16 analog input channels, the total time needed for AD conversion of all 16 channels was measured. The software driver was first written in C. This version needed 1000μsec to convert all 16 analog inputs to their binary representation. This amount of time is too long because the ADC rating, in manufacturer's technical specifications, is 80 kHz throughput when only one input channel is used. This is the ideal situation when the input multiplexer does not have to be used between subsequent conversions. It implies, however, that 12.5μsec is the shortest time needed for one AD conversion. The time achieved with the software driver written in C programming language, when all input channels were used, is 62.5μsec. This is more than 5 times the ideal ADC conversion time per one channel.

When the C driver acquired a stable form, it was reprogrammed in assembler. The time for all 16 AD conversions achieved by the assembler driver was 300μsec, i.e. 18.75μsec/channel. The ADC can be used in several modes, e.g. free run with auto-incrementing, external trigger mode, DMA transfer mode, overlap mode, and polling mode. The polling mode with individual addressing of each input channel was chosen because it is the fastest mode and most of the conversion and addressing process is under software control.

To test that the results from ADC are correct when all channels are converted sequentially, each odd input channel was connected to $+10V$ and each even input channel was connected to $-10V$. This test revealed that the the assembler driver was too fast and the measured results exhibited cross-talk effects from the previous channel. The reason was in timing performance of the ADC hardware because the timing of completion of each AD conversion, which is indicated by the end-of-conversion flag located in one of several Lab Master registers, is advancing the real end of the AD conversion. This necessitated implementation of delay loops after the test of the end-of-conversion flag. When the minimum wait loop was determined, the entire ADC for all 16 input channels took 600μsec, i.e. 37.5μsec/channel.

A.8 QDB Software Driver

The QDB keeps track of DD motor shaft positions from the instant it was reset. The QDB assembler software driver is in Appendix B.2 under the name *encoder.asm*. Its function, when called from the main program written in C, is to sample both counters holding a number of pulses proportional to the current shaft position and transfer the sampled 32 bit long integer number into the calling C program. The original design of QDB assumed that 16 bits could be read in parallel over the AT I/O bus. This effort was unsuccessful. The reason was that any I/O instruction addressing the I/O space is working with ports which hold only 8 bits. The QDB design was changed so that each eight bits of the sampled counter memory are connected to the D7-D0 bits of the I/O bus.

The original assembler driver used four IN instructions, i.e. one per byte. When all 4 bytes were received they were cascaded into 80386 32 bit extended registers and passed to the calling C program. Since the IN instruction of the 80386 CPU can also work with the extended AX (i.e.EAX) register a better solution is obtained because such IN instruction can read in 4 subsequent bytes from the I/O space because it autoincrements

Appendix B

Software Drivers - Source Code

B.1 Tecmar.c

```
/*  Name:       TECMAR.C
 *
 *  written by F.Niscak in May 1990
 *
 *  Function: subprograms for sampling which are used by
 *            all control programs running on the host PC computer
 *            and which mostly use the Labmaster Tecmar board
 *
 */

#include    "pc.h"
#include    "intface.h"

extern union REGS inregs,outregs;
extern struct SREGS segregs;
extern VECTOR old_vec;

extern unsigned int   far *vector;
extern unsigned int   far *data_seg;
extern unsigned long disbuf;
extern long count;
extern int int_mask;
extern int int_active;
extern int timer_units;
extern int load_reg;
```

```
/*********************************************************************/
/*  Return the current Code Segment register */
/*********************************************************************/
unsigned int cs()
{
    struct SREGS segregs;

    segread(&segregs);
    return(segregs.cs);
}

/*********************************************************************/
/*  Return the current Data Segment register */
/*********************************************************************/
unsigned int ds()
{
    struct SREGS segregs;

    segread(&segregs);
    return(segregs.ds);
}


/*********************************************************************
*  Initialization of the TECMAR controller
*********************************************************************
*/

void sampling_go()
{
    unsigned int counter =  5 ;      /* 1000 msec */

    outp(TIMER_CONTR, 0xff);  /* Master Reset of 9513 */
    outp(TIMER_CONTR, 0xe8);  /* Disable Data Pointer Sequencing */
    outp(TIMER_CONTR, 0x17);  /* Data Pointer to Master Mode Reg. */
    outp(TIMER_DATA, 0x10);   /* Low byte in Master Mode Register */
/* FOUT source = src1, both comparators = disabled, TOD = disabled */
    outp(TIMER_DATA, 0xd1);   /* High byte in Master Mode Register */
/* BCD division, disable data pointer incrementing, 8-bit bus,
 * FOUT = OFF, FOUT divider = 1 */
    outp(TIMER_CONTR, 0x01);  /* Data Pointer to Counter Mode Reg. */
    outp(TIMER_DATA, 0x25);   /* Low byte to Counter Mode Register */
/* disable special gates, reload from LOAD, count repetitively,
 * binary count, count down, active low terminal count pulse */
    outp(TIMER_DATA, timer_units);  /* High byte to Counter */
                                    /* Mode Register*/
/* no gating, counting on rising edge, count source ==
 * 0xb=1MHz, 0xc = 100 kHz, 0xd=10kHz, 0xe =1kHz, 0xf=100Hz */
```

```
    outp(TIMER_CONTR, 0x09);  /* Data Pointer to Load Register */
    outp(TIMER_DATA,load_reg&0x00ff); /* Low byte to Load Register */
    outp(TIMER_DATA,load_reg>>8);     /* High byte to Load Register */
    outp(TIMER_CONTR, 0x61);          /* Load and Arm Counter #1 */


    outp(TIMER_INT_ACKN,0);

    int_mask = inp(0x21);     /* Read the original 8259 IMR */

    outp(0x21,int_mask & (~(MASK_IRQ5 | MASK_KEYBOARD)) );
                              /* enable irq5 and keyboard */
    outp(0x20,0x20);          /* confirm to 8259 */

}


/*************************************************************************
*   Stopping of  the TECMAR controller
*************************************************************************
*/
void sampling_end()
{

    /* restore the original interrupt vector */
    *(vector+1) = old_vec.seg;
    *vector = old_vec.off;

    outp(TIMER_CONTR,0xff);   /* master reset of 9513 */
    outp(0x20, 0x20);         /* EOI for pending interrupt */

    /* resume normal priority setting */
    outp(0x21,int_mask);       /* restore the original 8259 IMR */
    outp(TIMER_INT_ACKN,0x00); /* clear TECMAR interrupt */

    /* generate zero torque signal on DA chan #0 */
    outp(TECMAR+1,0x0);       /* DA chan #0 = 0V, high byte */
    outp(TECMAR,0x0);         /* DA chan #0 = 0V, low byte */

}


/*************************************************************************
* This routine will initialize the IBM and the TECMAR.
*************************************************************************/
void init()
{
```

```
    data_seg  = FP_INT_ADDR(cs(), START - 2);
                                /* Address of that var */
    *data_seg = ds();           /* Save the current data segment*/

/* First get and SAVE the previous TECMAR timer interrupt vector */
    vector = FP_INT_ADDR(0, TIMER_IRQ); /* Vector addr */

    old_vec.seg = *(vector+1);
    old_vec.off = *vector;

    /* Setup vector for the Interrup Service Routine START */
    *(vector+1) = cs();         /* Segment of interrupt handler */
    *vector = (unsigned int) START; /* Offset of interrupt */

    outp(0x20,0x20);

    outp(TIMER_INT_ACKN,0x00); /* Clear TECMAR interrupt */
    outp(TECMAR+4,0x90);       /* TECMAR control byte */
    /* D7=1 => disable AD autoincrementing
       D4=1 => counter 9513 generates an
               interrupt pulse on jumper J7
     */

    outp(TECMAR+1,0x0);        /* DA chan #0 = 0V, high byte */
    outp(TECMAR,0x0);          /* DA chan #0 = 0V, low byte */

    sampling_go();             /* Initialize IRQ5 from 9513A*/
    outp(0x20,0x20);

    int_active = 0;            /* Initialize ISR flag */
}


/*********************************************************
* This is the interrupt handler routine. It is called *
* by the assembler interrupt receiving routine.       *
*********************************************************/
void handler()
{
    count++;
    int_active = 1;            /* ISR was active */
}


/*************************************************************
```

```
* Procedure: TIMERGO
* Function : Initializes, clears and starts Tecmar counter #5.
* Input Frequency: 1 MHz
* Maximum count: 65 537 usec
* To stop : call the the function TIMEREND
*****************************************************************/
void  timergo()
{
    outp(TECMAR+9, 0xff);  /* Master Reset of 9513 */
    outp(TECMAR+9, 0xe8);  /* Disable Data Pointer Sequencing */
    outp(TECMAR+9, 0x17);  /* Data Pointer to Master Mode Reg. */
    outp(TECMAR+8, 0x10);  /* Low byte into Master Mode Register */
    outp(TECMAR+8, 0xd1);  /* High byte into Master Mode Register */
    outp(TECMAR+9, 0x05);  /* Data Pointer to Counter Mode Reg. */
    outp(TECMAR+8, 0x2d);  /* Low byte to Counter Mode Register */
    outp(TECMAR+8, 0x0b);  /* High byte to Counter Mode Register */
    outp(TECMAR+9, 0x0d);  /* Data Pointer to Load Register */
    outp(TECMAR+8, 0x00);  /* Low byte to Load Register */
    outp(TECMAR+8, 0x00);  /* High byte to Load Register */
    outp(TECMAR+9, 0x70);  /* Load and Arm Counter #5 */
}


/*****************************************************************
*  Procedure: TIMEREND
*Function : Stops the counter #5 and prints its content
*****************************************************************/
void timerend()
{
    outp(TECMAR+9, 0x90);  /* Disarm and Save Counter #5 to Hold */
    outp(TECMAR+9, 0x15);  /* Data Pointer to Hold Register #5 */
    printf("\nMicroseconds = %u",
                          ((inp(TECMAR+8)<<8) | inp(TECMAR+8)));
}
```

B.2 Encoder.asm

```
    .MODEL  SMALL
    .386

    .DATA
MOTOR1 DW  203H ;Port address of the highest byte of motor #1 counter
MOTOR2 DW  207H ;Port address of the highest byte of motor #2 counter
SAMPLE DW  201H ;Port address for SAMPLING
DWORD1 DW  200H ;Port address of the LOWEST  byte of motor #1 counter
```

```
DWORD2 DW  204H ;Port address of the LOWEST  byte of motor #2 counter


    .CODE
PUBLIC _encoder

_encoder    PROC
        push    bp          ;Entry sequence - save old BP
        mov     bp, sp      ;Set stack framepointer

        mov     dx, SAMPLE  ;Sampling of both counters
        out     dx, al

        mov     bx, [bp+6]  ;BX = ADDRESS of the long COUNT2
                            ;in C program

        mov     dx, DWORD2  ;MOTOR #2
        in      eax, dx     ;Double word
        neg     eax         ;GET PROPER DIRECTION
        mov     [bx], eax   ;Write COUNT2 back to the
                            ;C calling program


        mov     bx, [bp+4]  ;BX = ADDRESS of the long COUNT1
                            ;in C program

        mov     dx, DWORD1  ;MOTOR #1
        in      eax, dx     ;Double word
        neg     eax         ;GET PROPER DIRECTION
        mov     [bx], eax   ;Write COUNT2 back to the
                            ;C calling program

        pop     bp          ;Exit sequence - restore old BP
        ret
_encoder    ENDP
            END
```

B.3 IRQ5.asm

```
; STATIC NAME ALIASES
;
TITLE IRQ5.ASM
NAME RQ5
```

```
;   FUNCTION: THIS ROUTINE WILL TRAP THE HARDWARE
; INTERRUPT REQUEST 5 AND WILL  CONFIRM:
;     1) 8259 IC <=> IRQ5 HAS BEEN PROCESSED
;     2) TECMAR 9513 TIMER <=> TIMER INT
;                     HAS BEEN PROCESSED

;              BOARD: LAB MASTER  -  TECMAR

               .MODEL SMALL
               .386


.CODE
TIMER_INT_ACKN  DW  717H ; TECMAR INTERRUPT ACKNOWLEDGE PORT ADDRESS
_DATA_SEG       DW  (0)  ; STORAGE OF SEGMENT ADDRESS


_START  PROC    FAR
        PUBLIC  _START
        EXTRN   _handler:NEAR
                                ;
        PUSHF                   ; Save the flags
        PUSHA                   ; Save all registers
        PUSH    DS              ;
        PUSH    ES              ;
                                ;
        MOV     AX,CS:_DATA_SEG ; Address our paramaters
        MOV     DS,AX
        CLC                     ; Clear the carry flag
        CLD                     ; Set direction flag forward
        CALL    _handler        ; Call 'C' interrupt handler
                                ;
        MOV     AL,20H          ; Standard interrupt return procedure
        OUT     20H,AL          ; confirmation to the 8259
                                ;
        MOV     DX,TIMER_INT_ACKN ;Confirmation of the interrupt to
        OUT     DX,AL           ; the Tecmar timer 9513A
                                ;
        POP     ES              ; Restore all registers
        POP     DS
        POPA
        POPF
                                ;
        IRET                    ; Interrupts on with return
_START ENDP
        END
```

B.4 ADC.asm

```
        TITLE   ADC.ASM
        NAME    ADC


        .MODEL  SMALL
        .386

        .DATA
TECMAR      DW      710H        ; Tecmar base address
NONINDEX    DW      714H        ; port adr for nonindexing mode
CHAN_NUM    DW      715H        ; port adr for chan_# to convert
END_ADC     DW      714H        ; port adr to test end of ADC
START_ADC   DW      716H        ; port adr to start ADC
ADC_RESULT  DW      715H        ; port adr to read result of ADC
DELAY       DW      2           ; delay loop counter


        .CODE

        PUBLIC  _adc
_adc    PROC    NEAR

        enter   WORD PTR  0,0 ; Procedure stack and allocation
;       enter   WORD PTR 12,0
        push    si

; Stack locations of parameters and variables
;       result = 4
;       first_chan = 6
;       chan_counter = 8

; The following section which initializes NONINDEXING MUST
; be performed by program which initializes also the Tecmar
; 9513A Counter/Timer
;       mov     dx, NONINDEX
;       mov     al, 80H
;       out     dx, al              ; initialize for nonindexing

        jmp     DECR_COUNTER

CHANNEL_LOOP:
        mov     ax, WORD PTR [bp+6] ; first_chan
        add     ax, WORD PTR [bp+8] ; chan_counter
```

```
        mov     dx, CHAN_NUM
        out     dx, al              ; write channel # to convert

        mov     dx, START_ADC
        out     dx, al              ; start ADC

        mov     dx, END_ADC
        xor     ax, ax              ; clear AX
READY_LOOP:
        in      al, dx              ; read status of ADC
        cmp     ax, 80H             ; test of status of bit 7
        jl      READY_LOOP


        mov     cx, DELAY           ; DELAY LOOP
DELAY_LOOP:
        loop    DELAY_LOOP
                                    ; ADC has completed
;       mov     dx, ADC_RESULT
        mov     dx, 715H
        in      ax, dx              ; read the result
        mov     bx,WORD PTR [bp+8]  ; chan_counter
        shl     bx,1                ; multiply * 2 to get byte address
        mov     si,WORD PTR [bp+4]  ; result
        mov     [bx][si],ax         ; save result

DECR_COUNTER:
        dec     WORD PTR [bp+8]     ; chan_counter
        jns     CHANNEL_LOOP

        pop si
        leave
        ret

_adc    ENDP
        END
```

Appendix C

Schematic Diagrams - Orcad

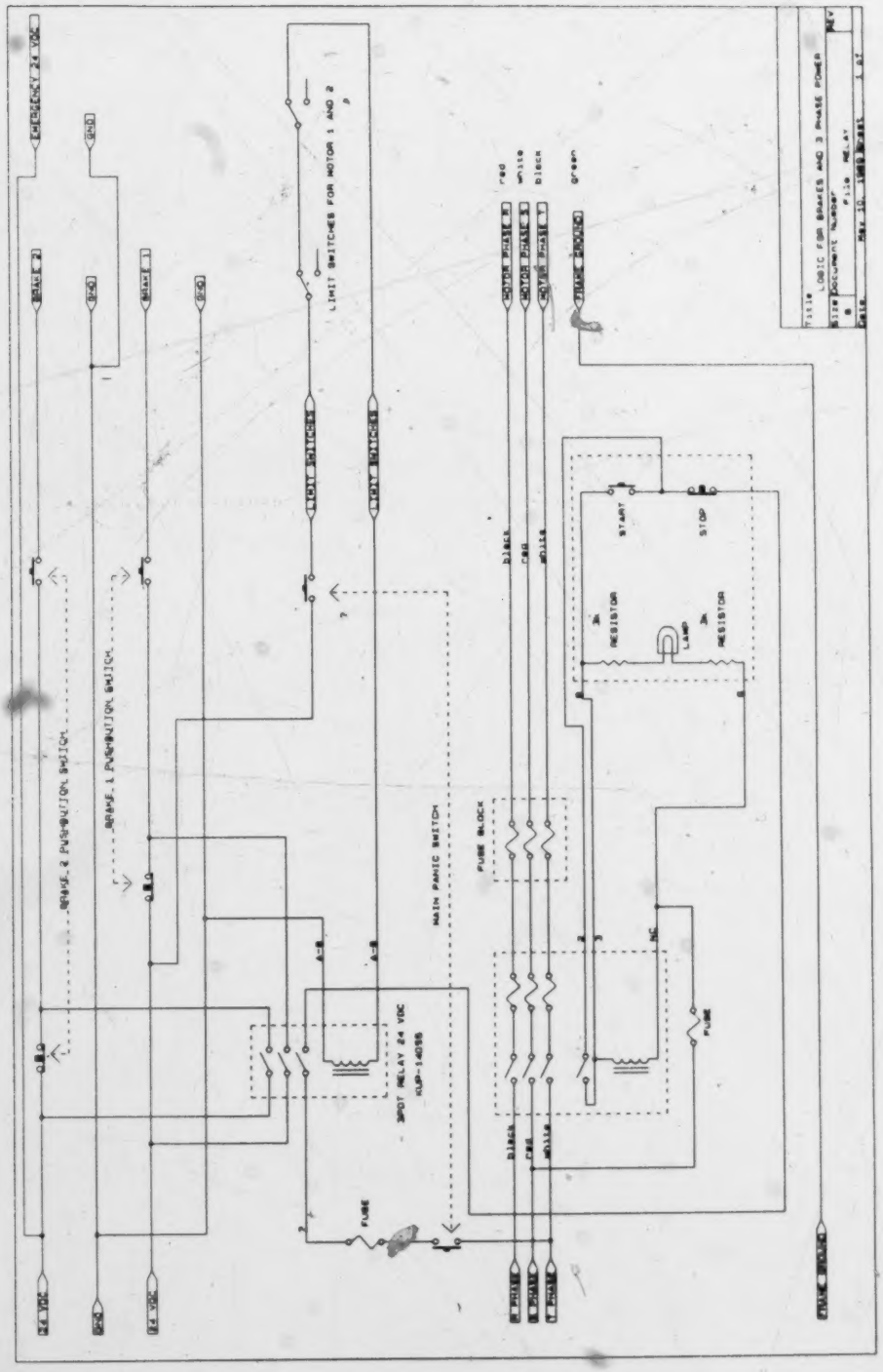


Figure C.1: Logic for brakes and 3 phase power.

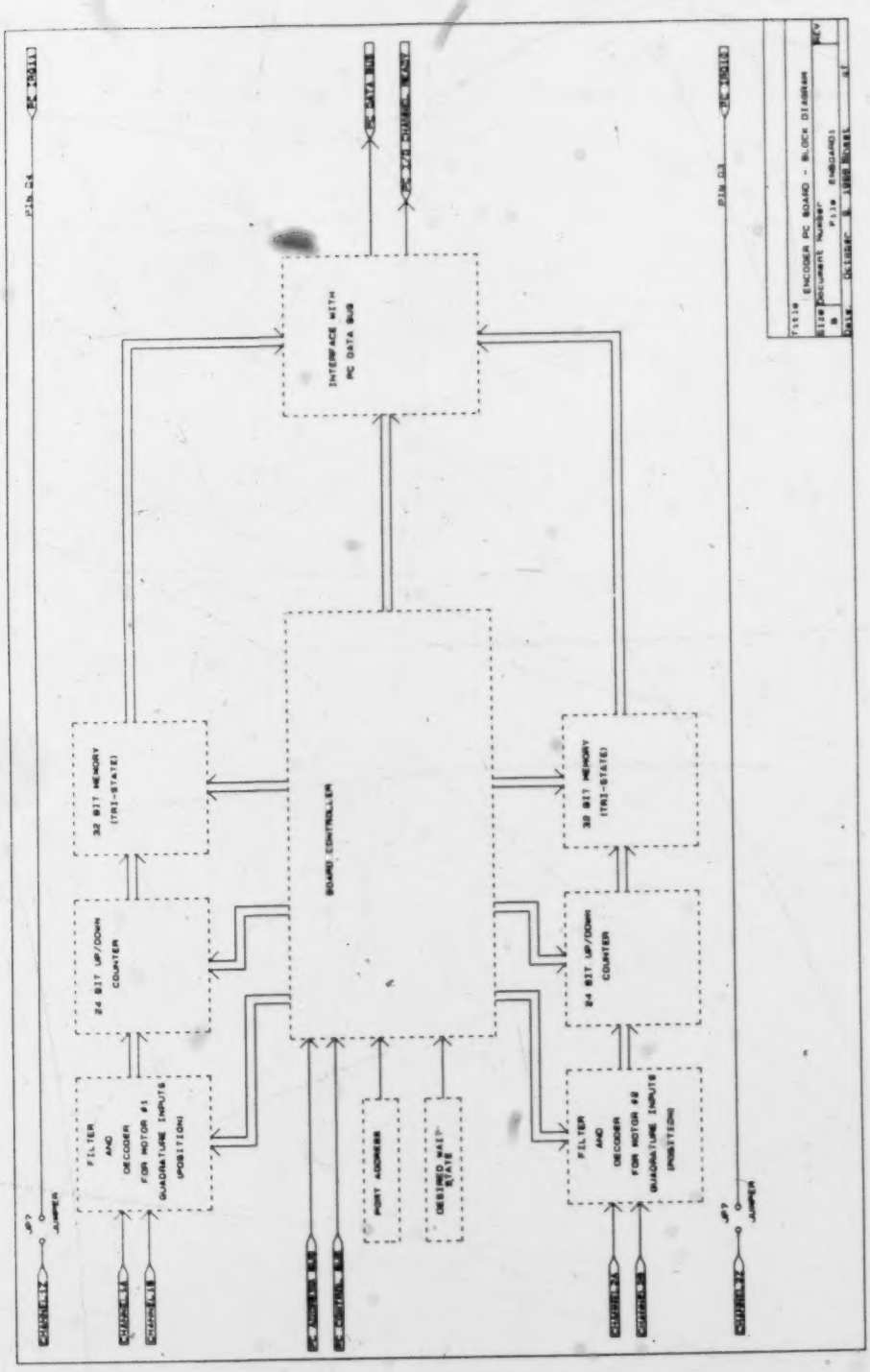

INTERFACE WITH PC DATA BUS
BOARD CONTROLLER
32 BIT MEMORY (TRI-STATE)
24 BIT UP/DOWN COUNTER
FILTER AND DECODER FOR MOTOR #1 QUADRATURE INPUTS (POSITION)
32 BIT MEMORY (TRI-STATE)
24 BIT UP/DOWN COUNTER
FILTER AND DECODER FOR MOTOR #2 QUADRATURE INPUTS (POSITION)
PORT ADDRESS
DESIRED WAIT STATE
JUMPER
JUMPER
ENCODER PC BOARD - BLOCK DIAGRAM

Figure C.2: Block diagram of the QDB.

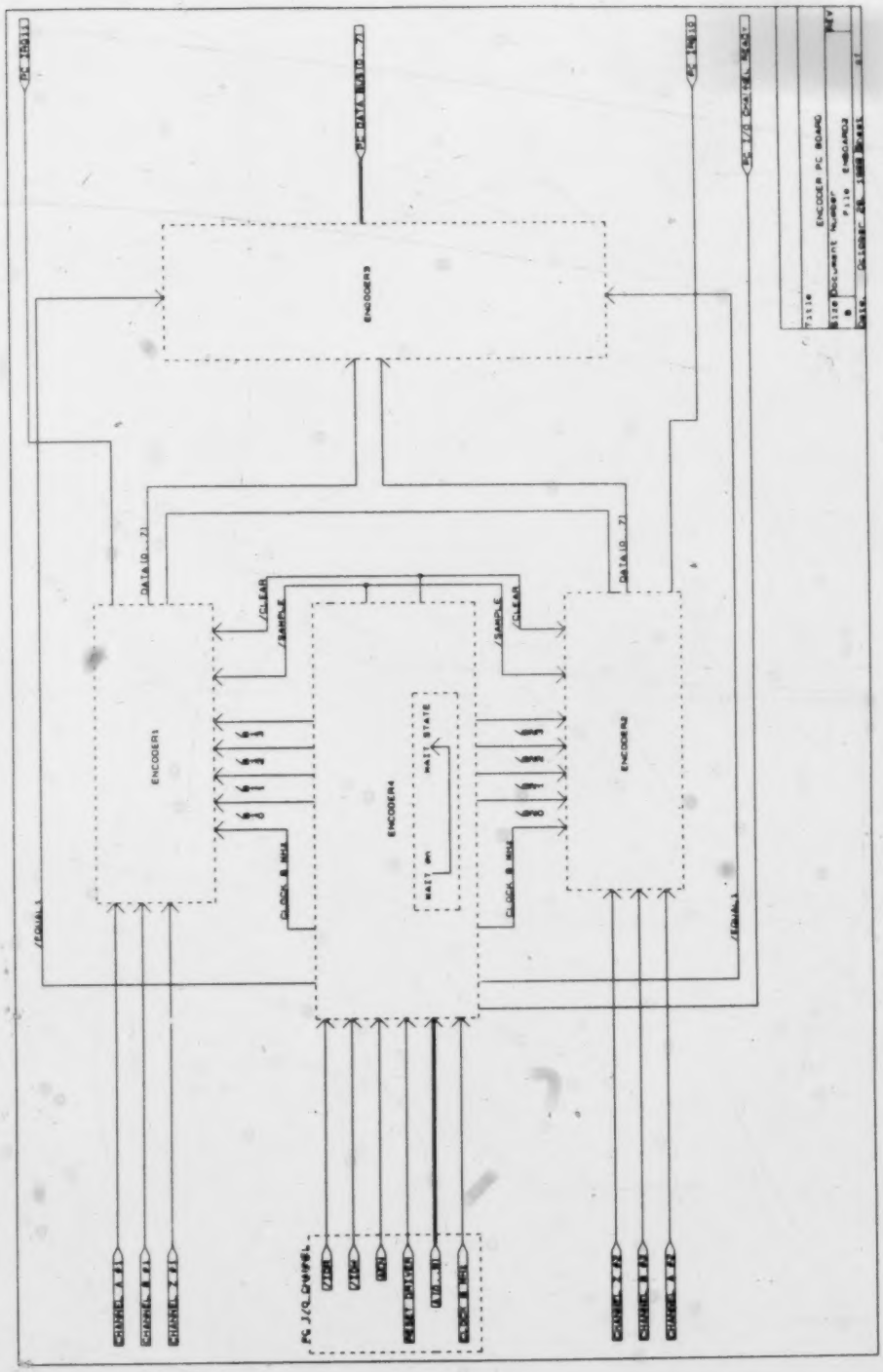


Figure C.3: Interconnections among basic QDB subsystems.

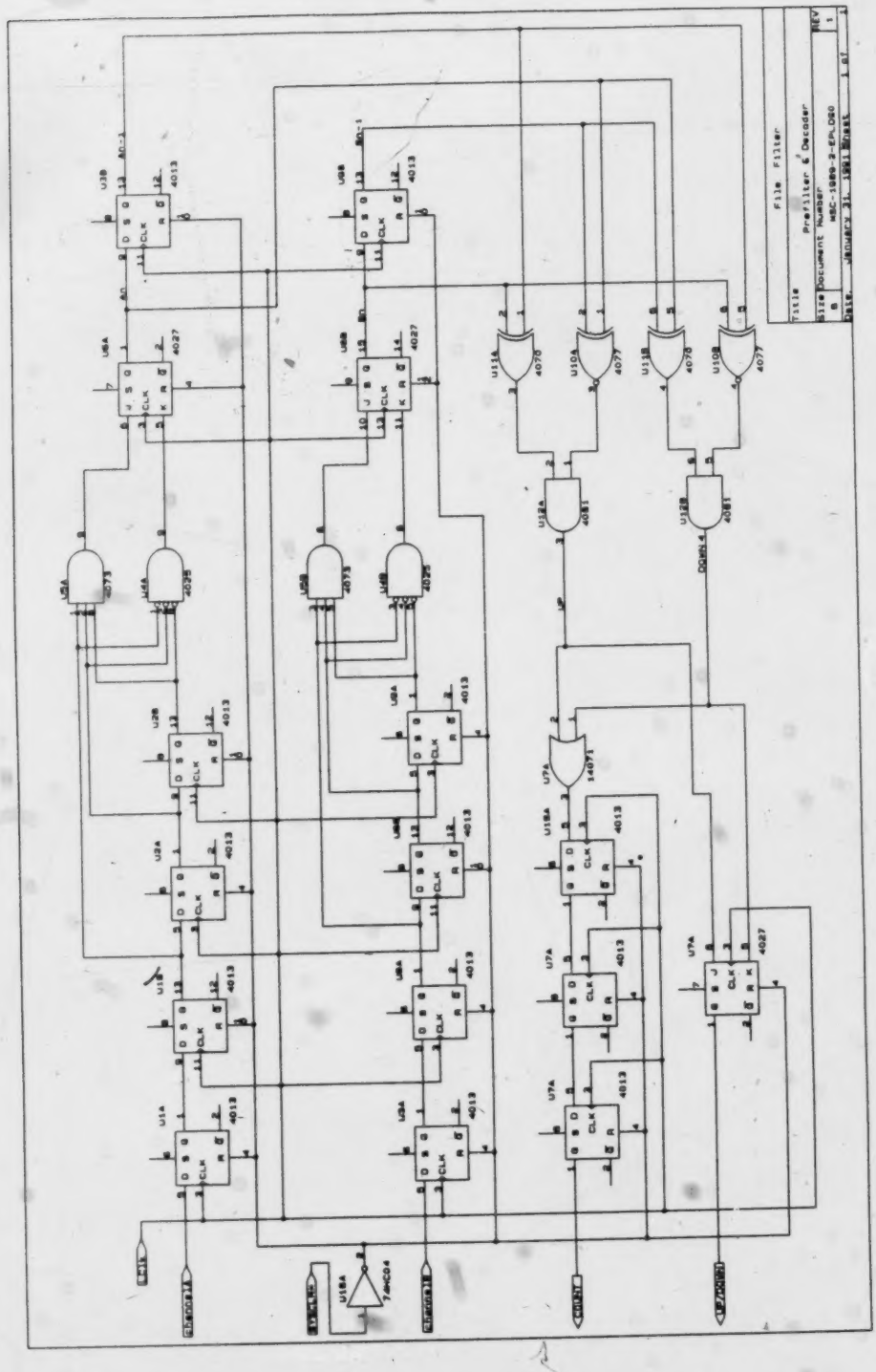


Figure C.4: Prefilter and decoder.

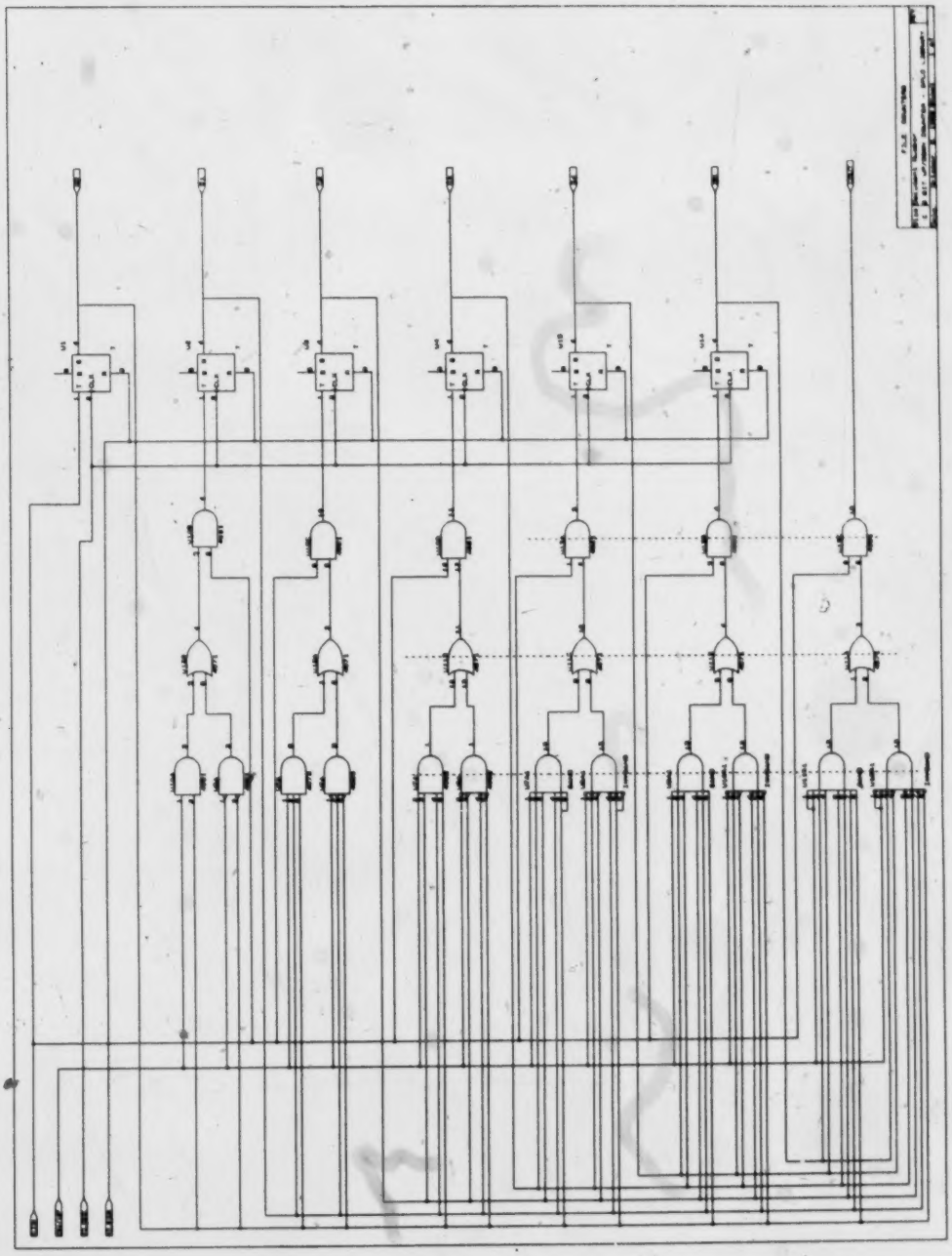


Figure C.5: 6-bit up/down Counter - EPLD library.

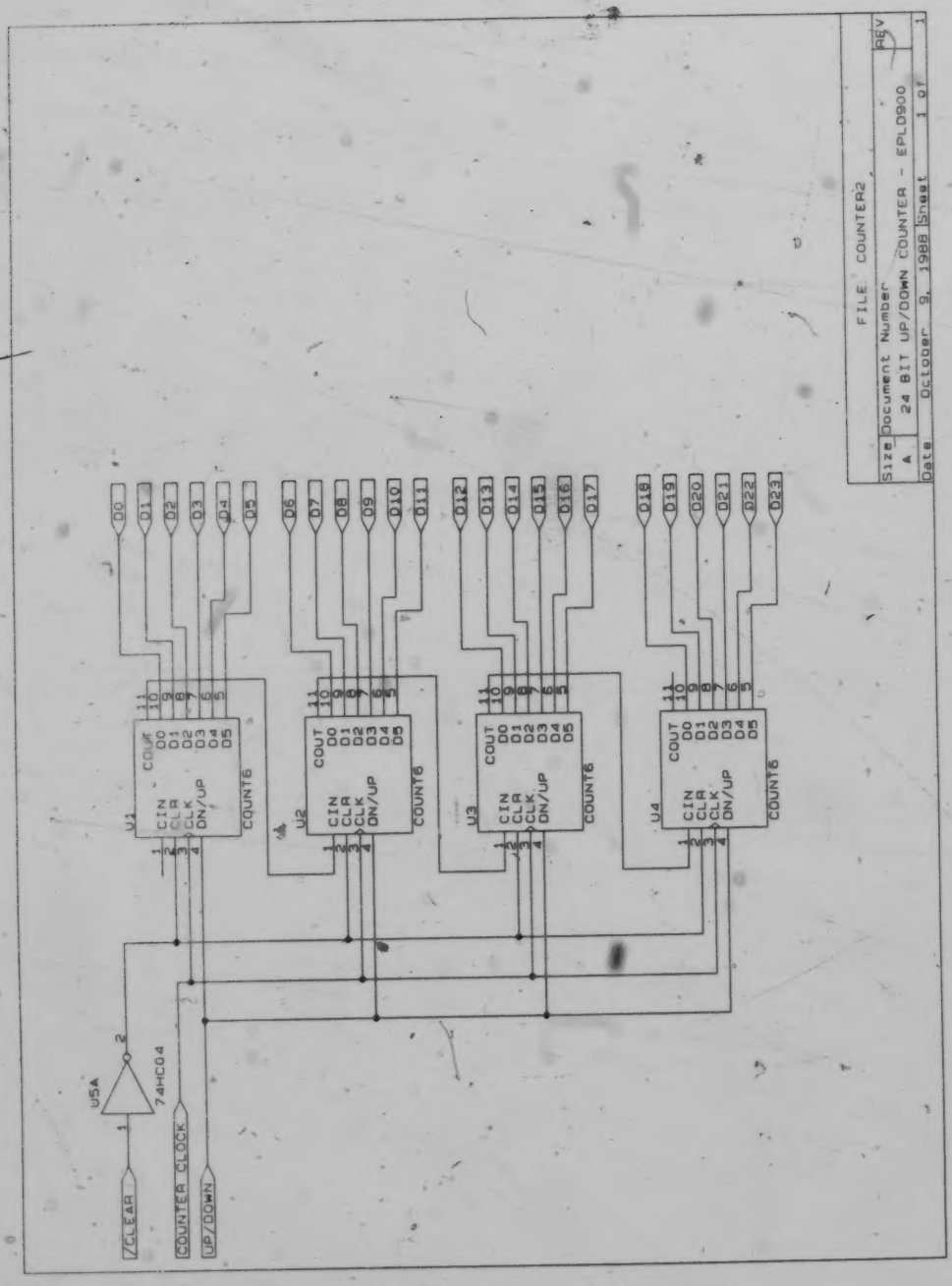

FILE COUNTER2
Size Document Number Rev
A 24 BIT UP/DOWN COUNTER - EPLD900
Date October 9, 1988 Sheet 1 of 1
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20 D21 D22 D23
U1 U2 U3 U4
COUNT6
COUT D0 D1 D2 D3 D4 D5
CIN CLR CLK DN/UP
U5A
74HC04
ZCLEAR
COUNTER CLOCK
UP/DOWN

Figure C.6: 24-bit up/down counter - EPLD900.

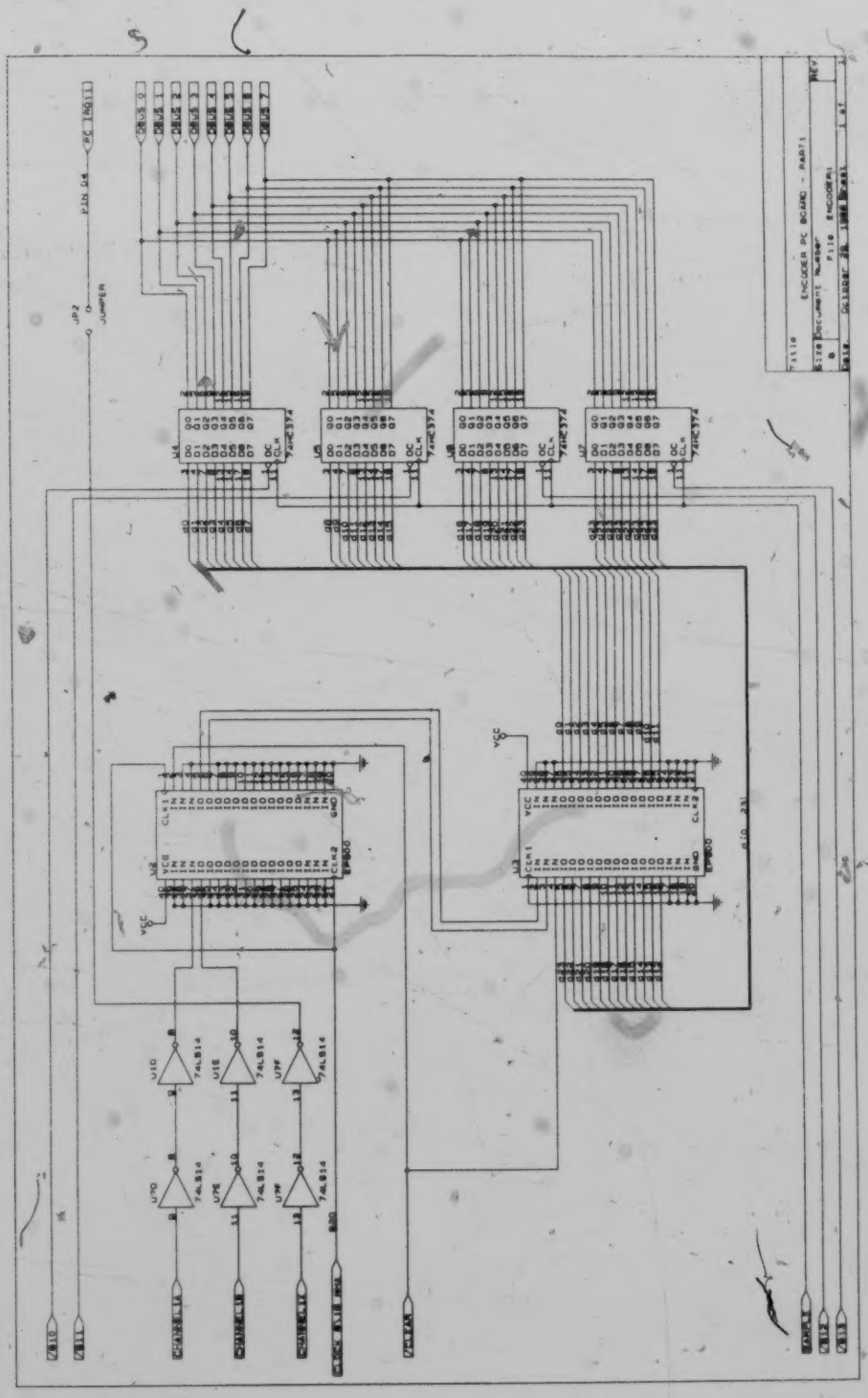


Figure C.7: Counter memory and I/O bus drivers - Part 1.

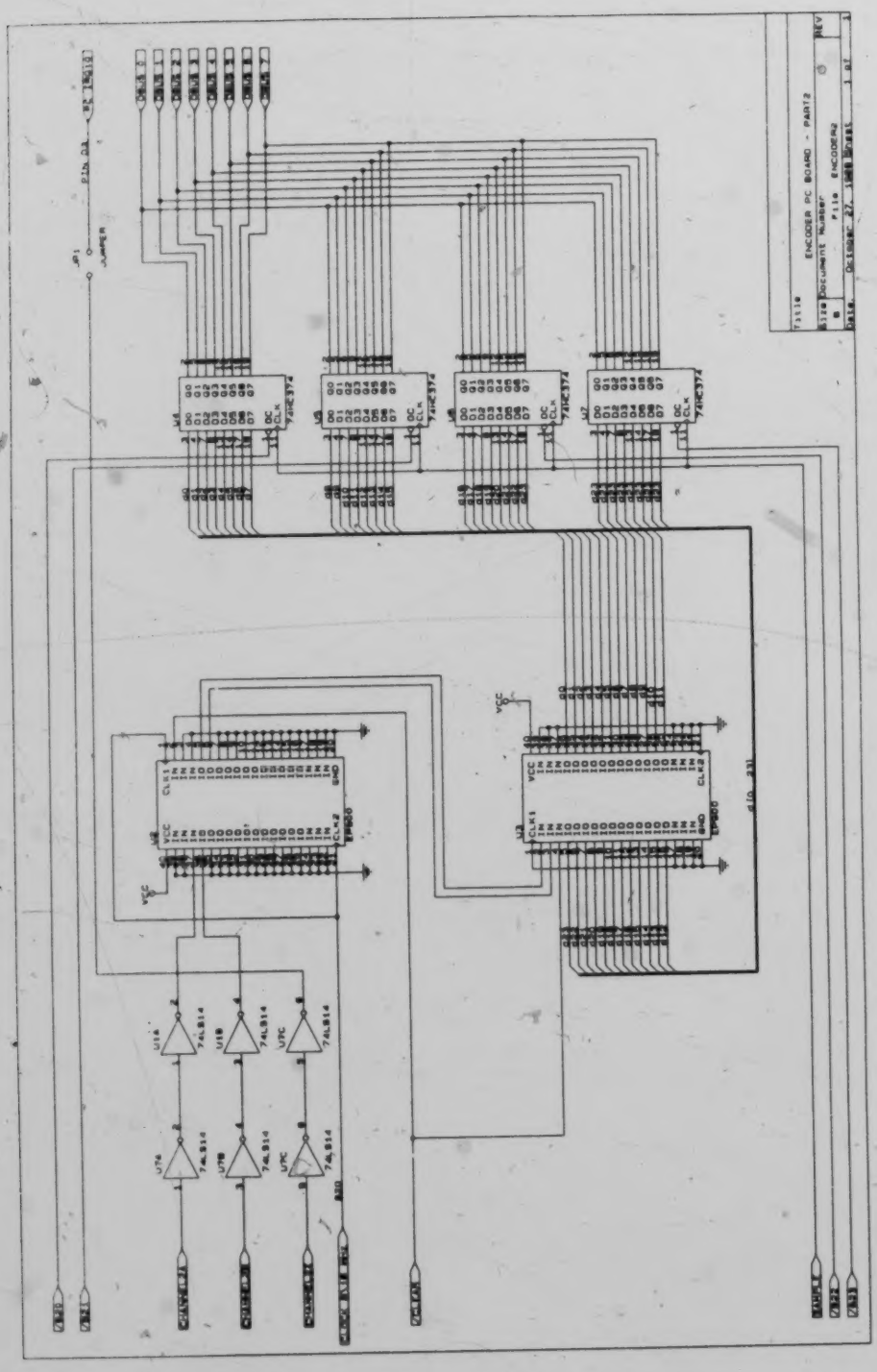


Figure C.8: Counter memory and I/O bus drivers - Part 2.

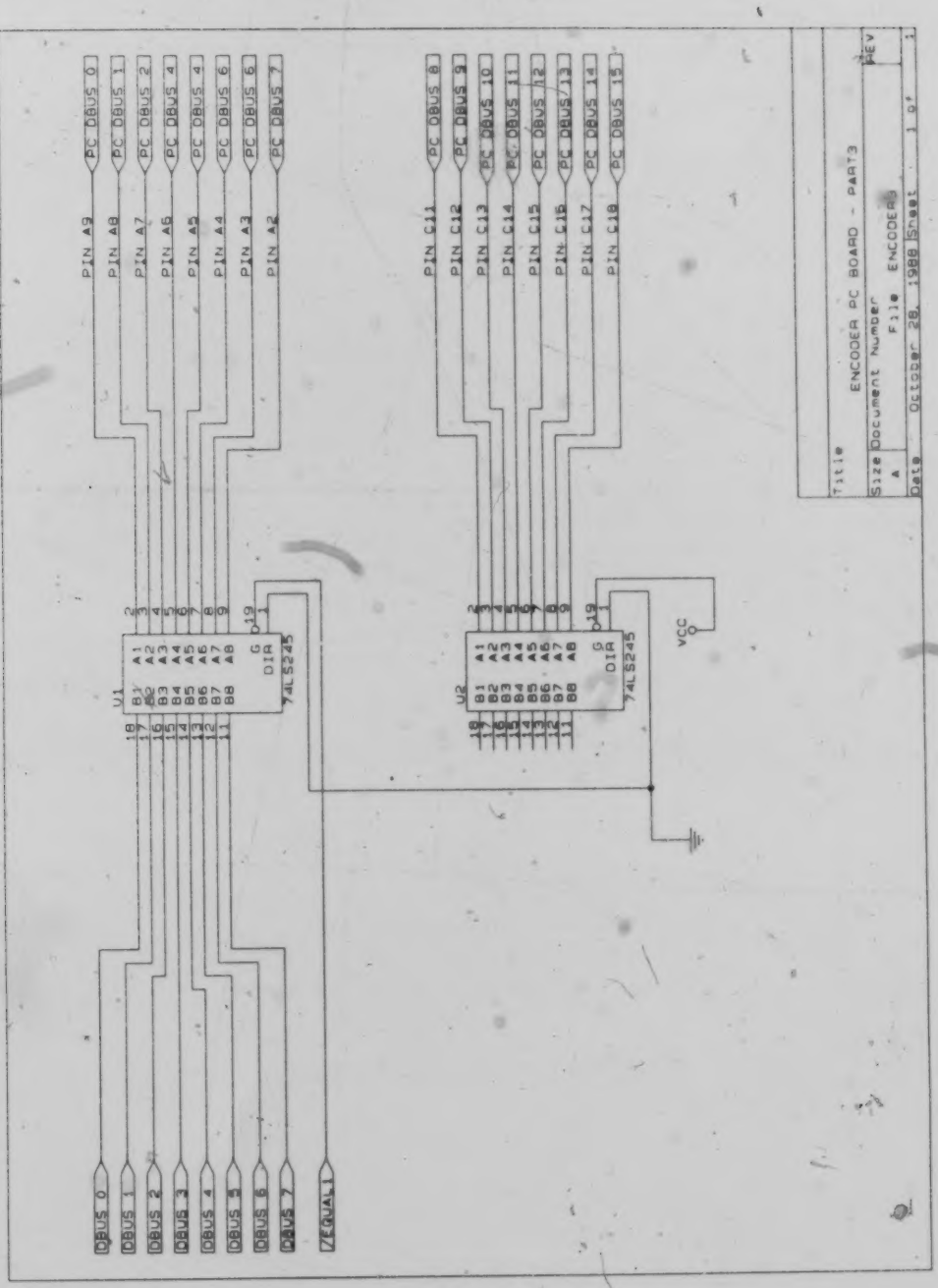


Figure C.9: Counter memory and I/O bus drivers - Part 3.

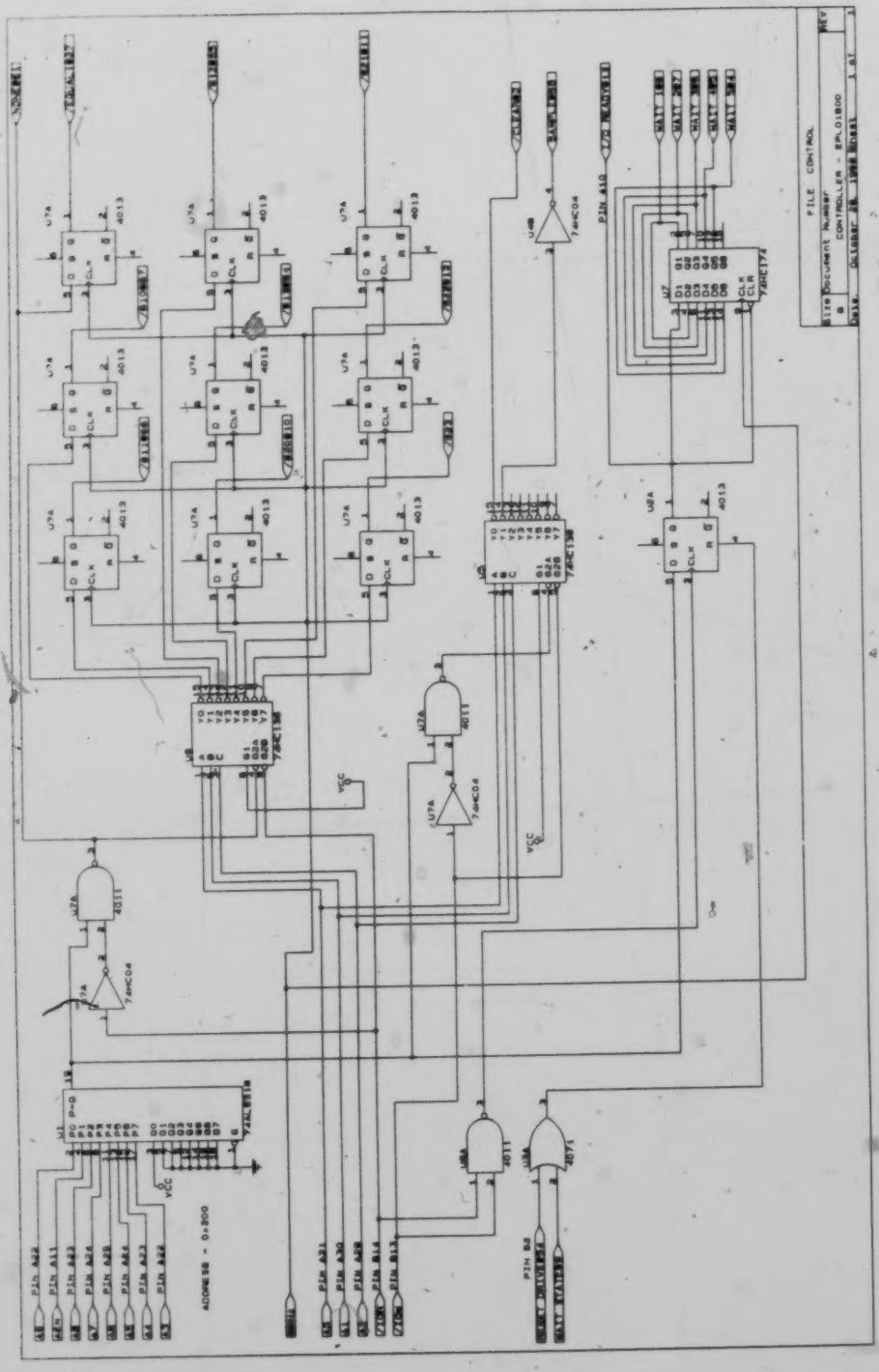


Figure C.10: QDB controller - internal structure.

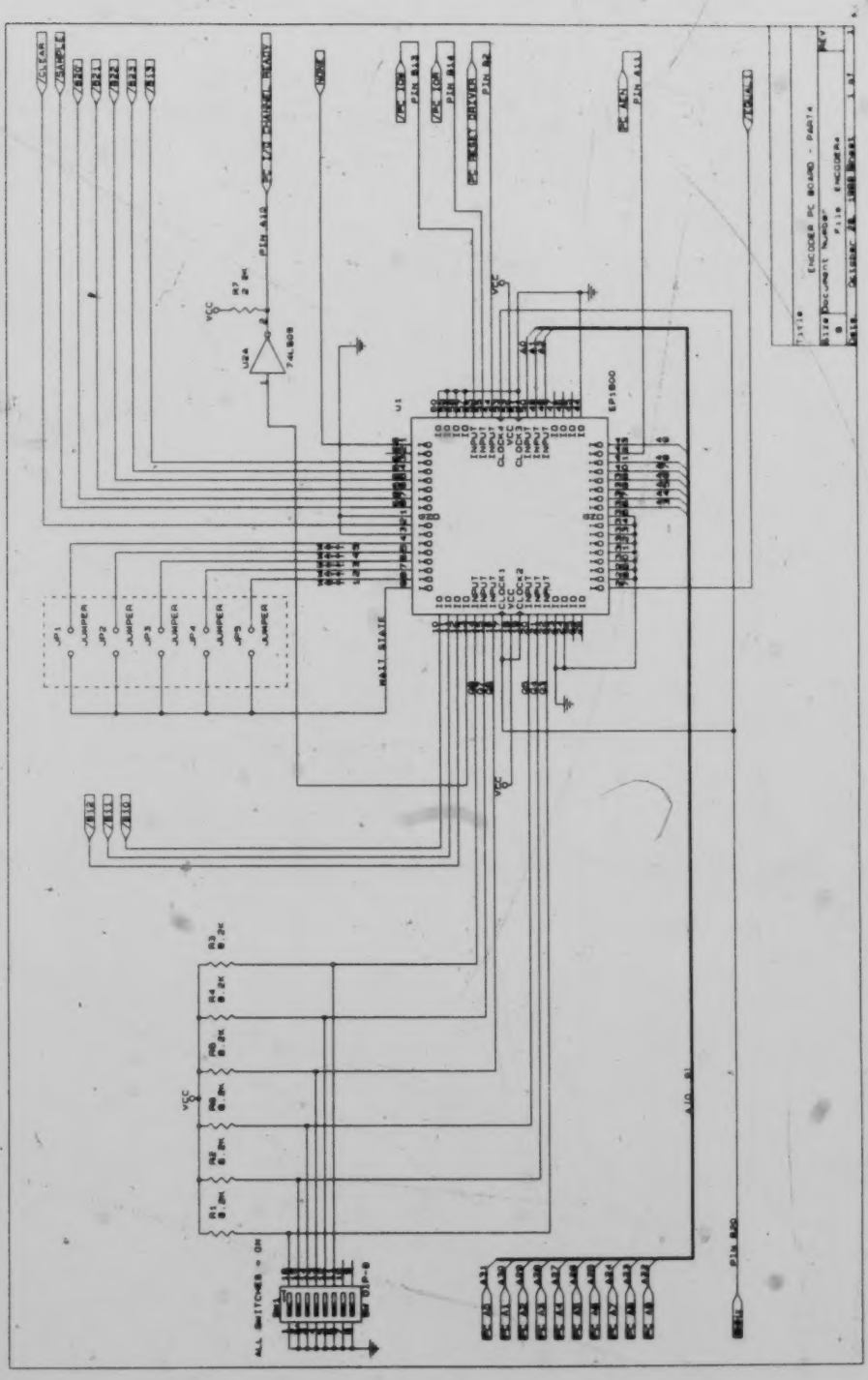


Figure C.11: QDB controller - external structure.

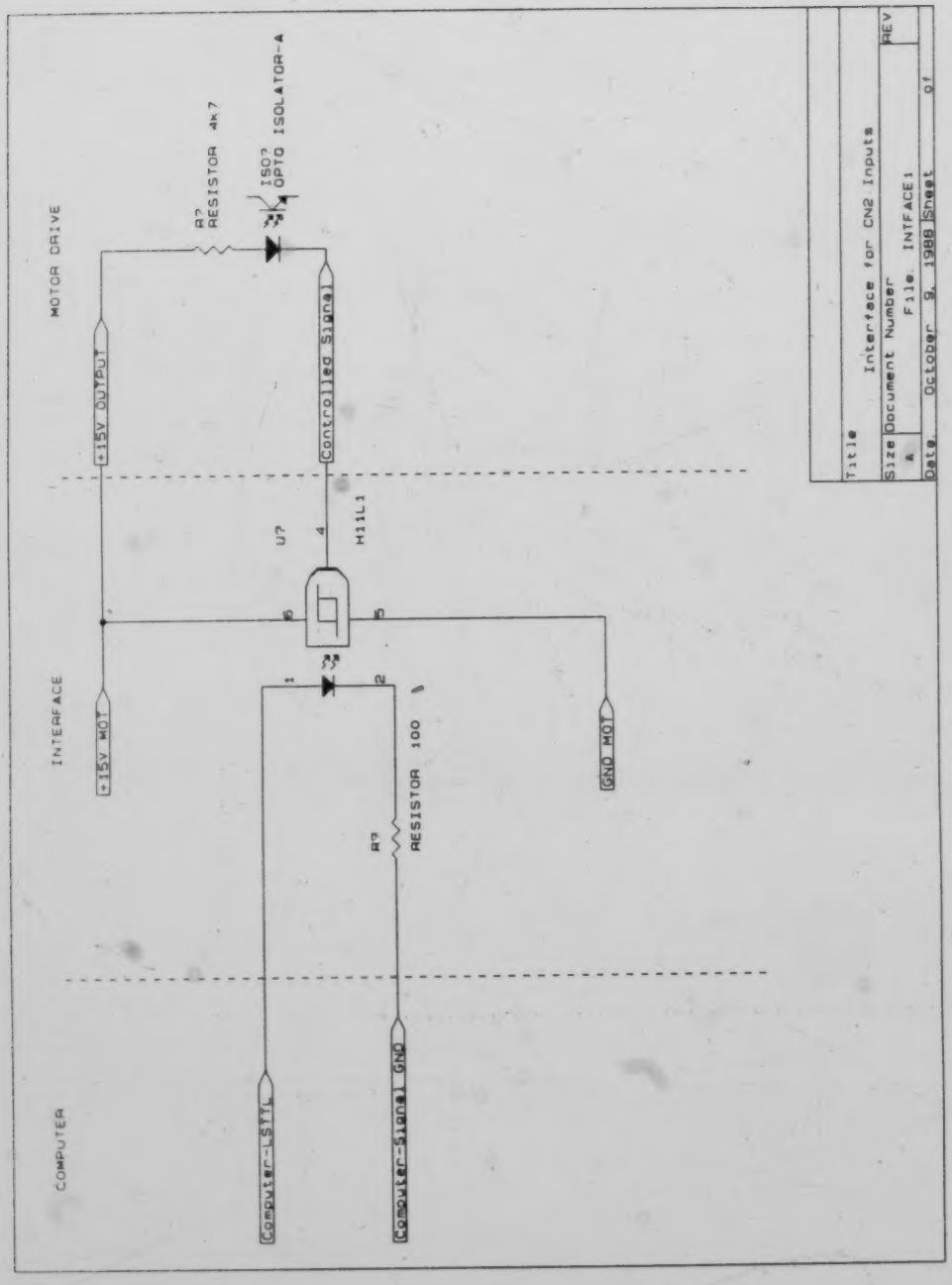


Figure C.12: Interface for CN2 inputs.

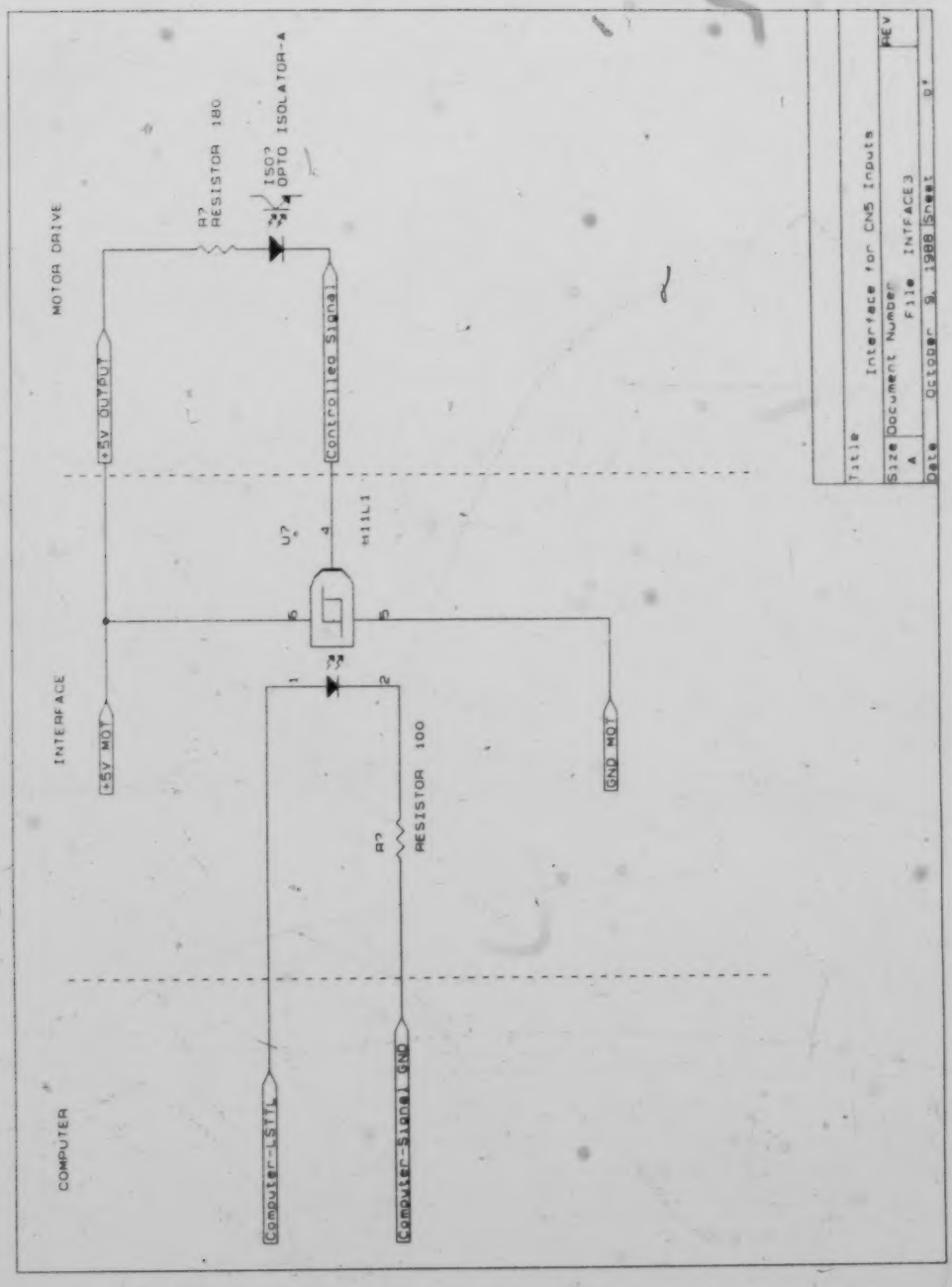


Figure C.13: Interface for CN5 inputs.

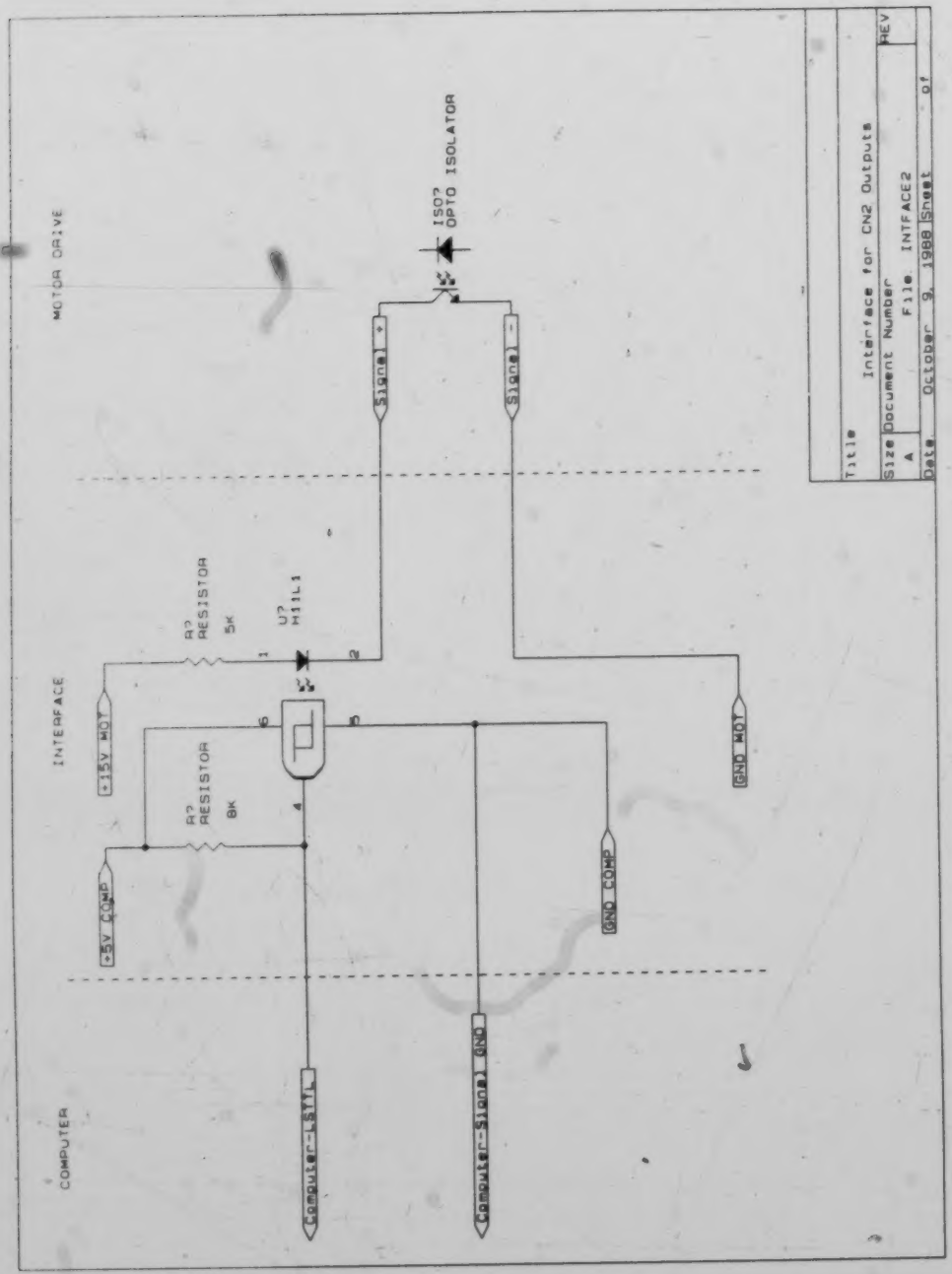


Figure C.14: Interface for CN2 outputs.

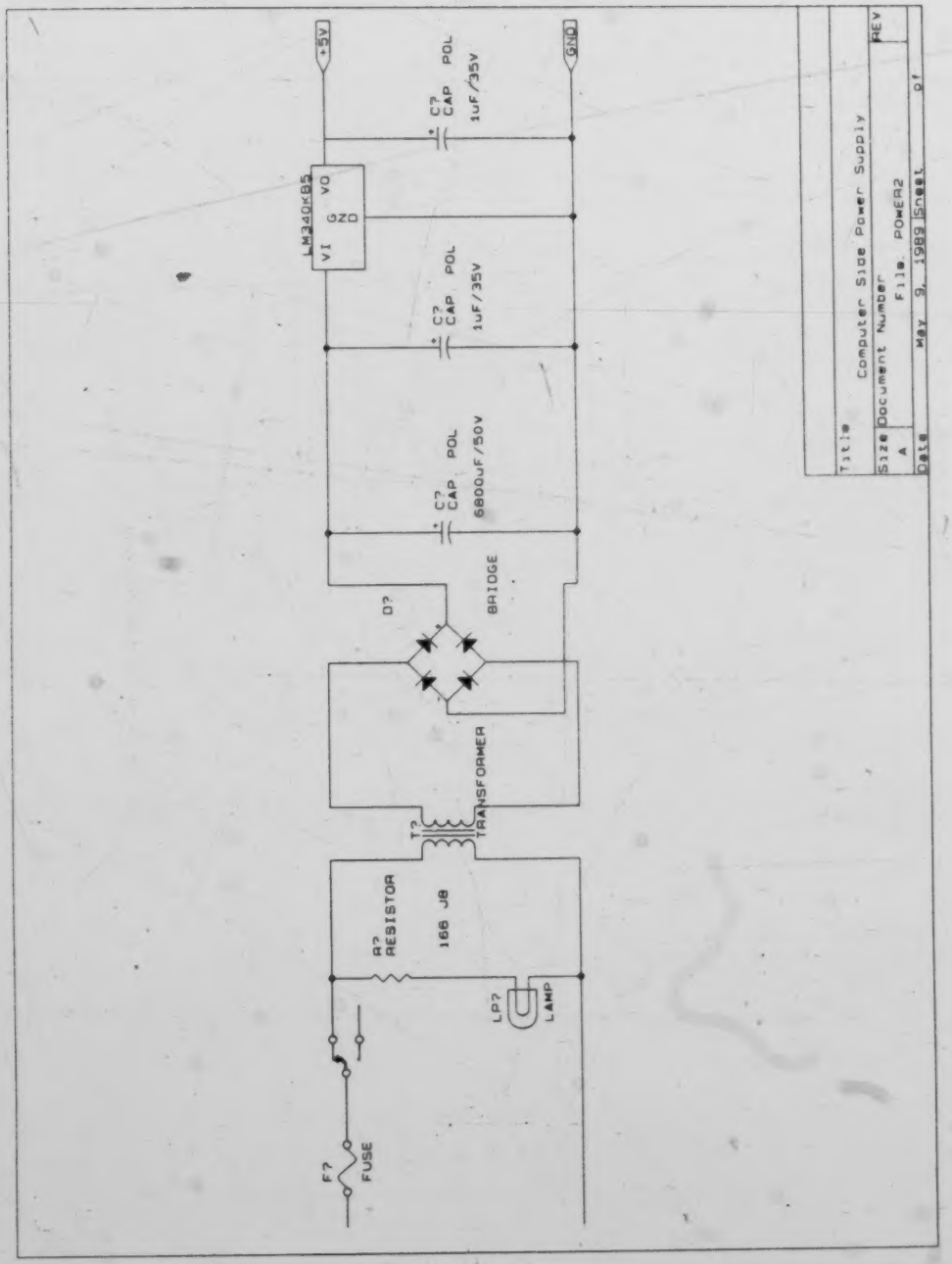


Figure C.15: Computer side power supply for the DI/O box.

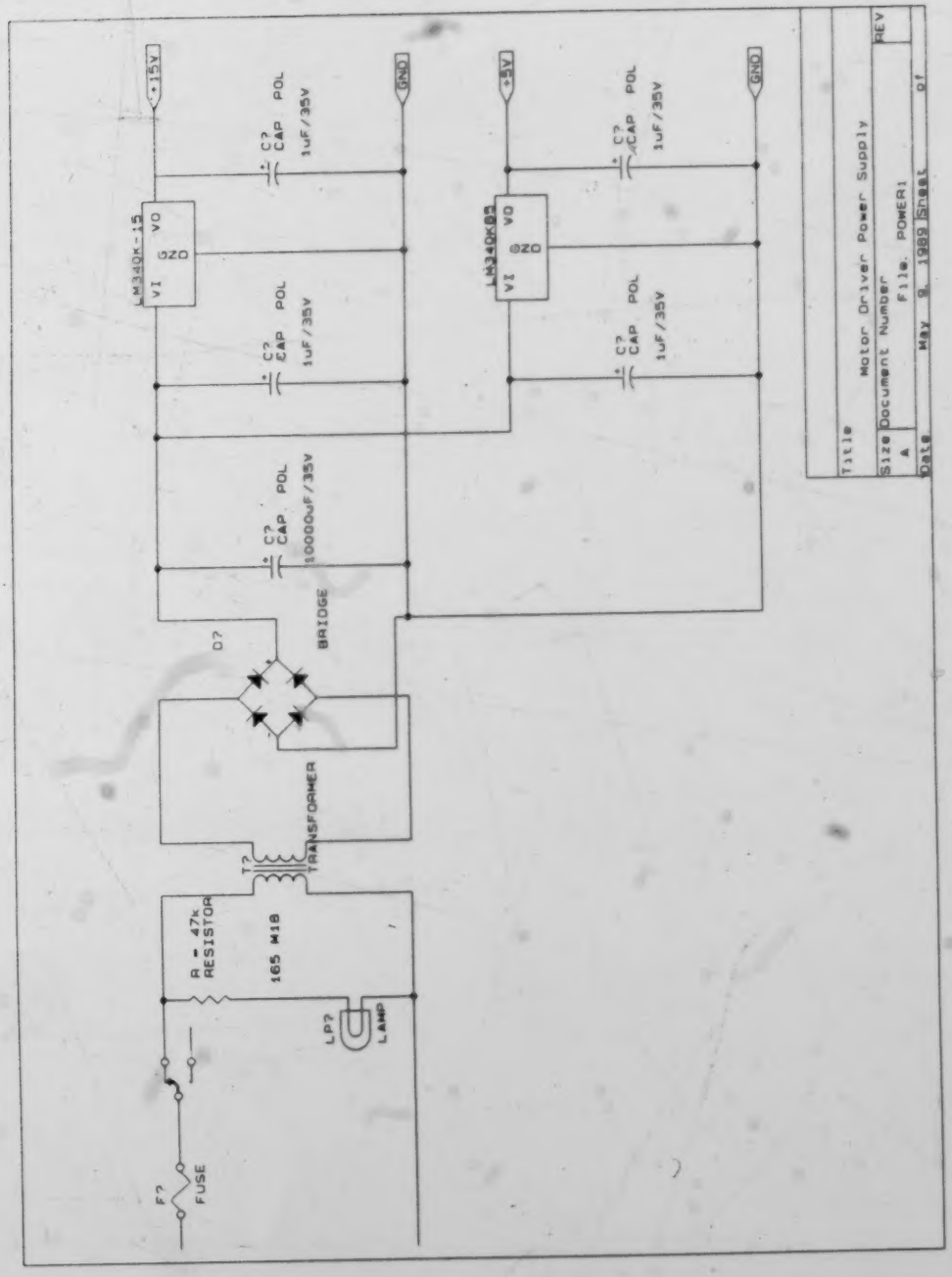


Figure C.16: Motor side power supply for the DI/O box.

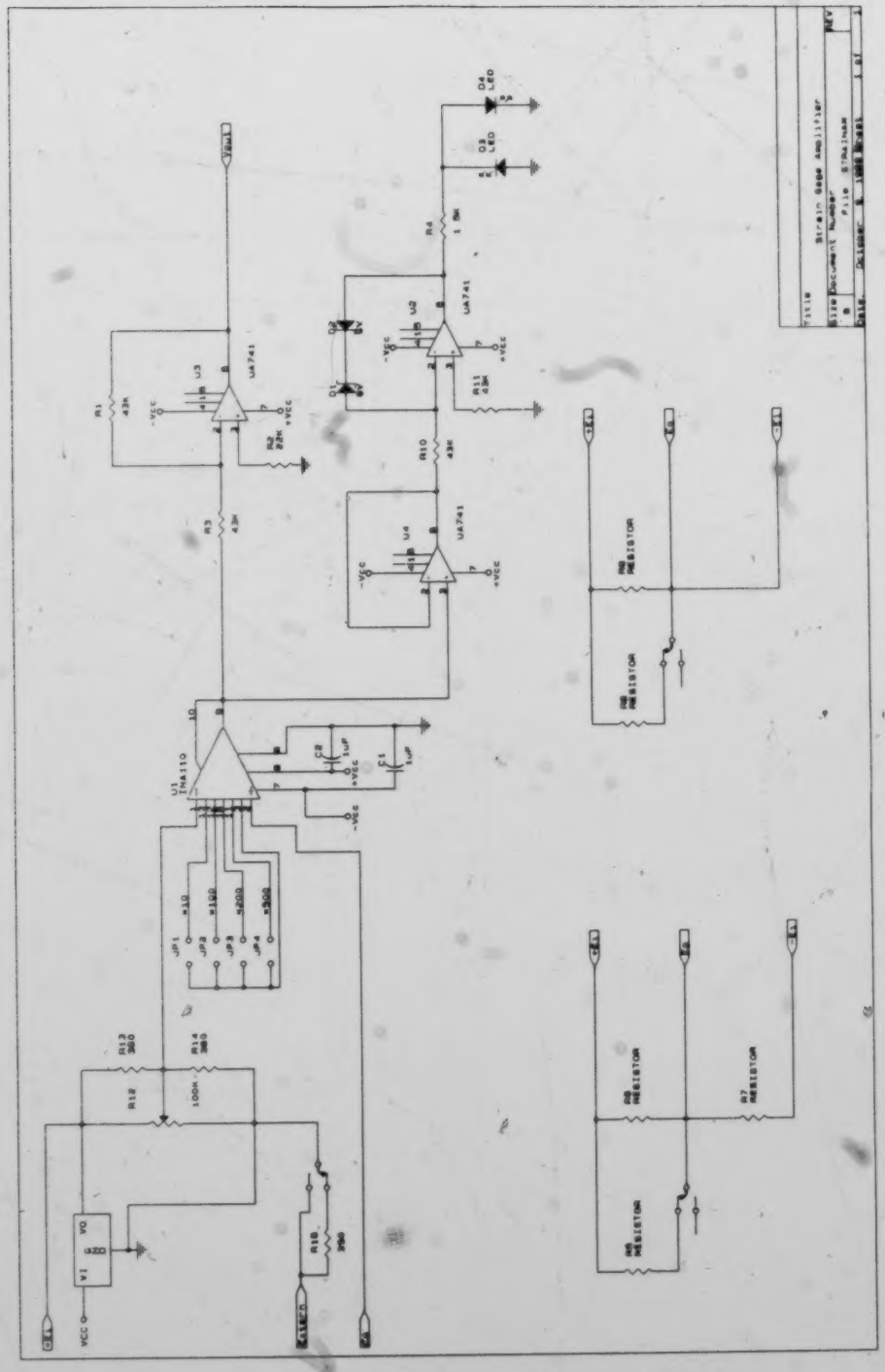


Figure C.17: Strain gage amplifier.

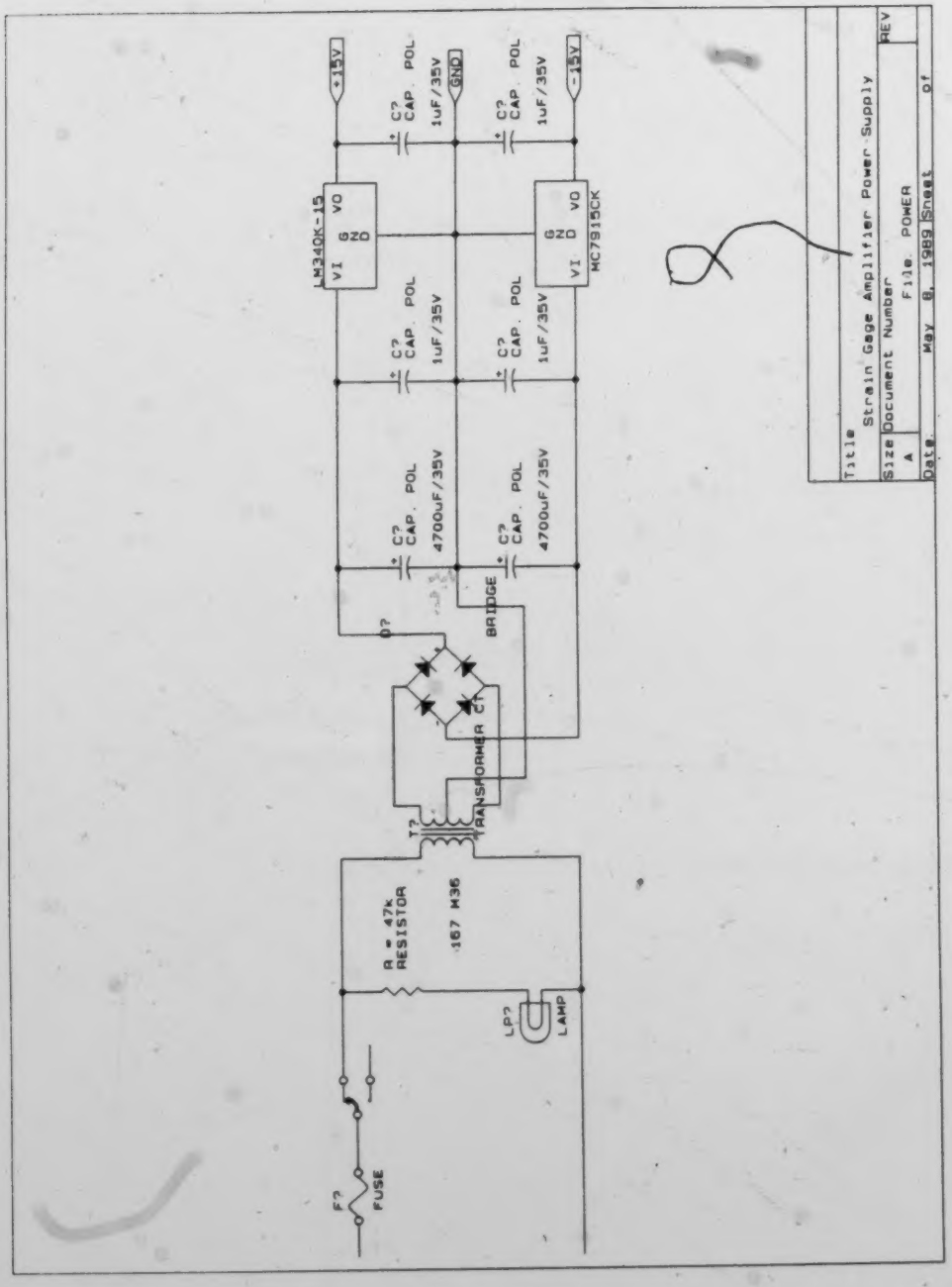


Figure C.18: Power supply for the strain gage amplifier.

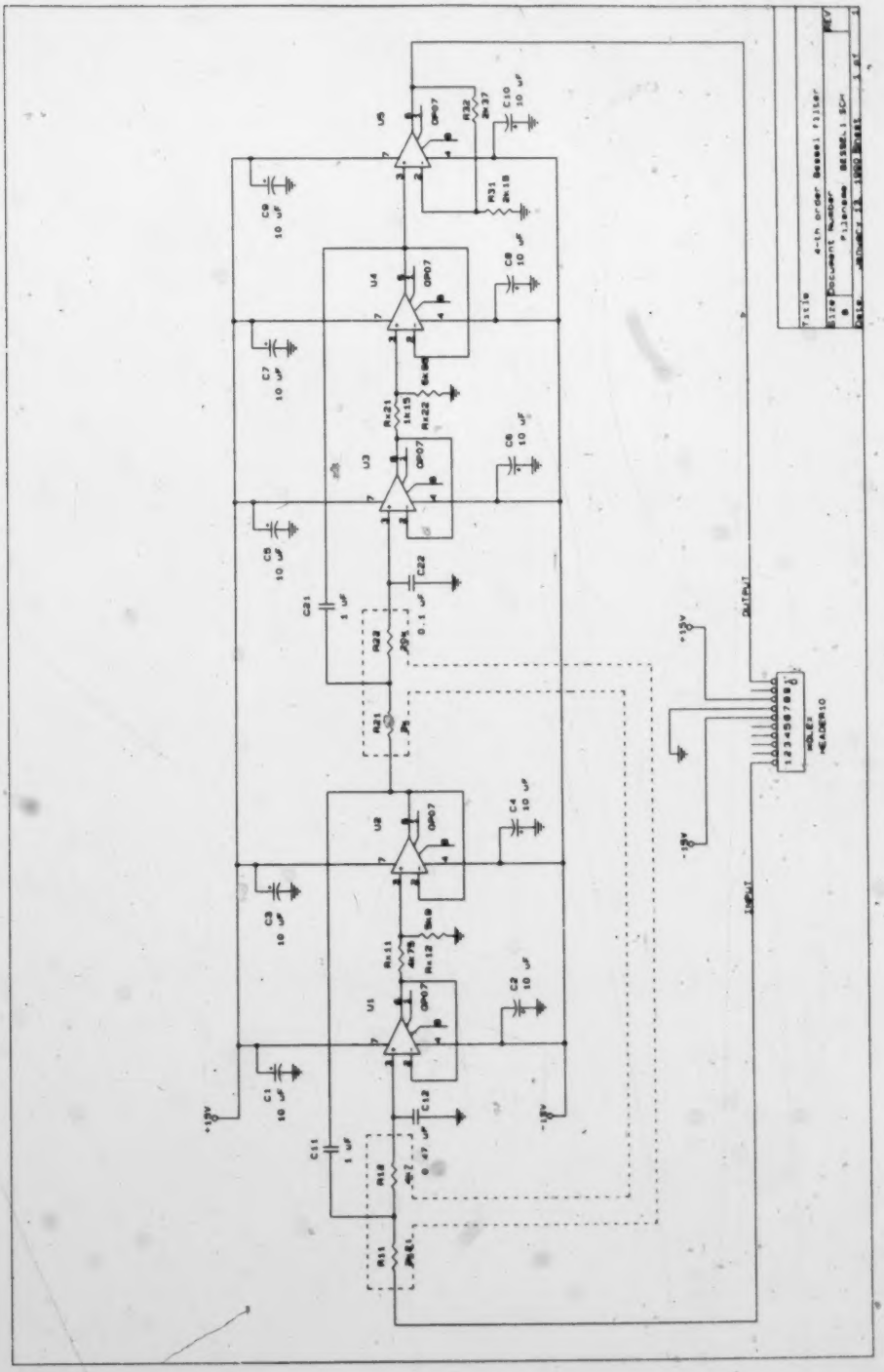


Figure C.19: The 4-th order Bessel antialiasing filter.

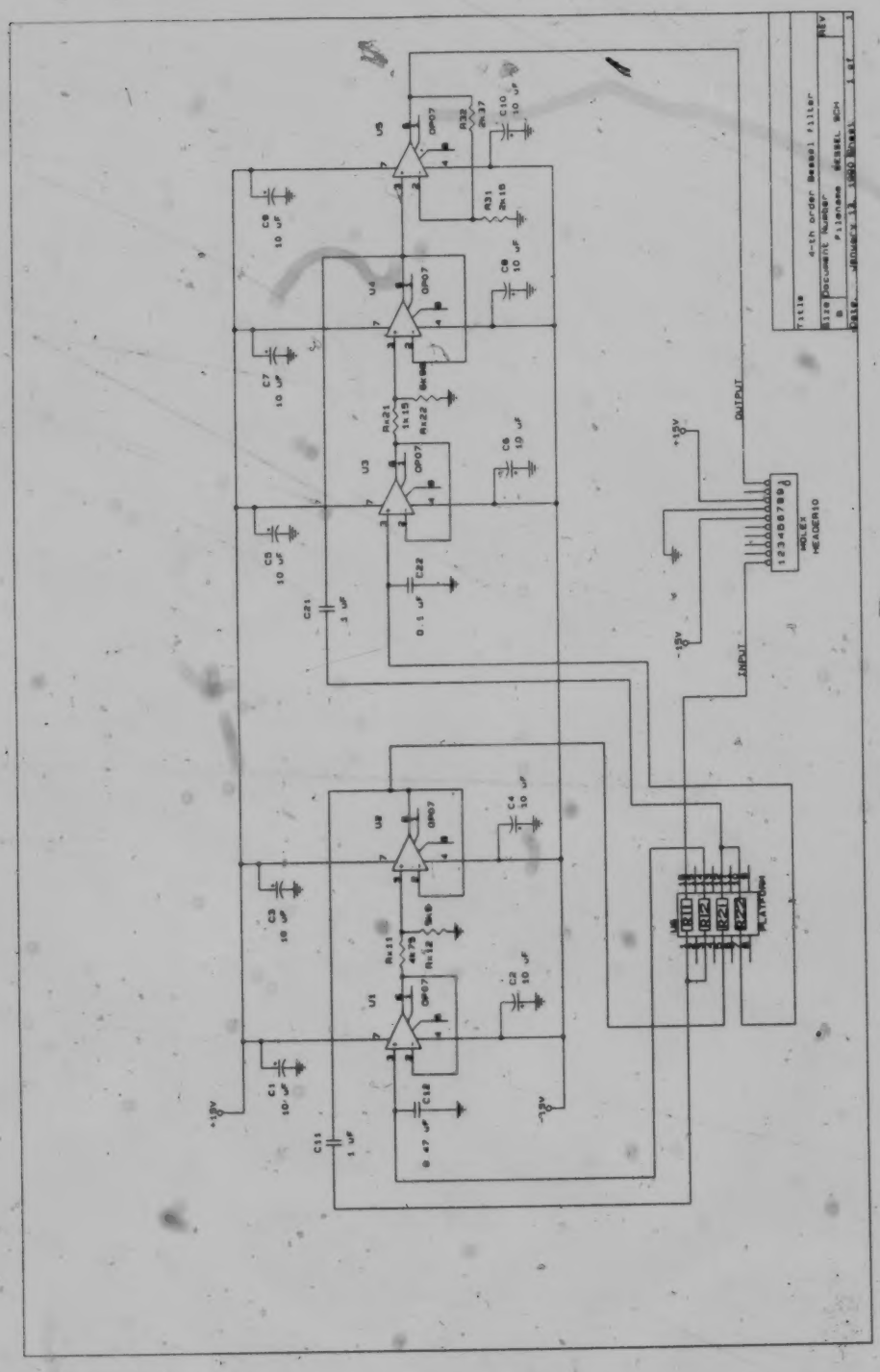


Figure C.20: The 4-th order Bessel filter with component carier.

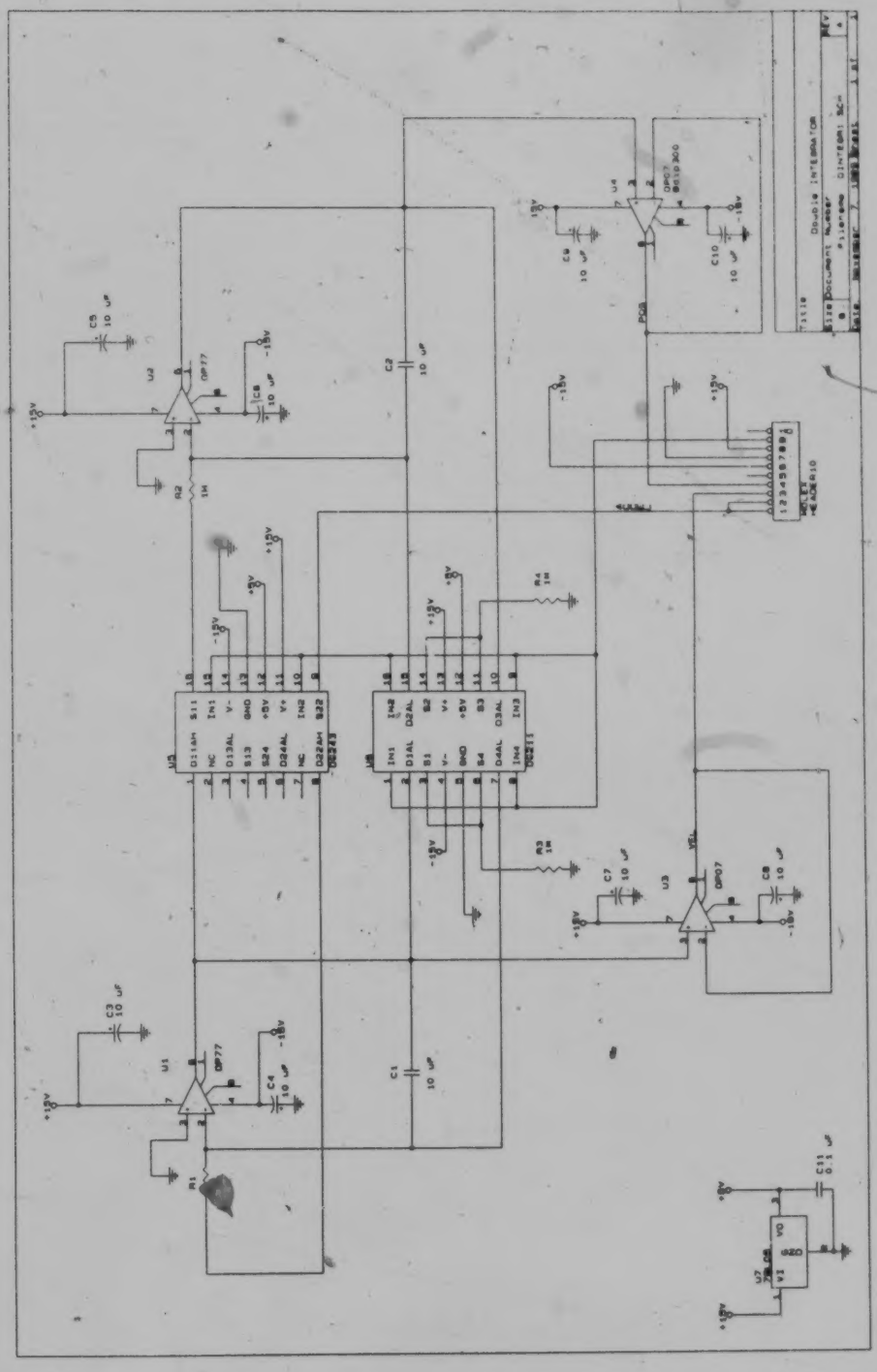


Figure C.21: Double integrator.